WebSphere Transcoding Publisher Version 4.0
Sharpening the Edge of the Network

Marshall Lamb
Chief Programmer, WTP
IBM, Research Triangle Park, NC

Transcoding at the Edge of the Network

What is the edge?

A “network” defines, in the technical sense, a communication medium between two or more computing devices. In reality, networks are typically complicated combinations of several smaller networks with many devices managing the bridges between them. The Internet, for example, contains hundreds of subnetworks and thousands of devices managing the routing of data. An edge in the network is any boundary between these subnetworks, the data sources, and the clients accessing that data. So you can imagine that in terms of identifying the edge of the network, there are many possibilities.

Figure 1 gives a simple view of a network with all of its edges. There is the edge between the data center and the various networks (intranet and Internet), then between the networks themselves, and finally between the networks and the end-user devices. Deploying WebSphere Transcoding Publisher (WTP) involves identifying which edge it is to serve.

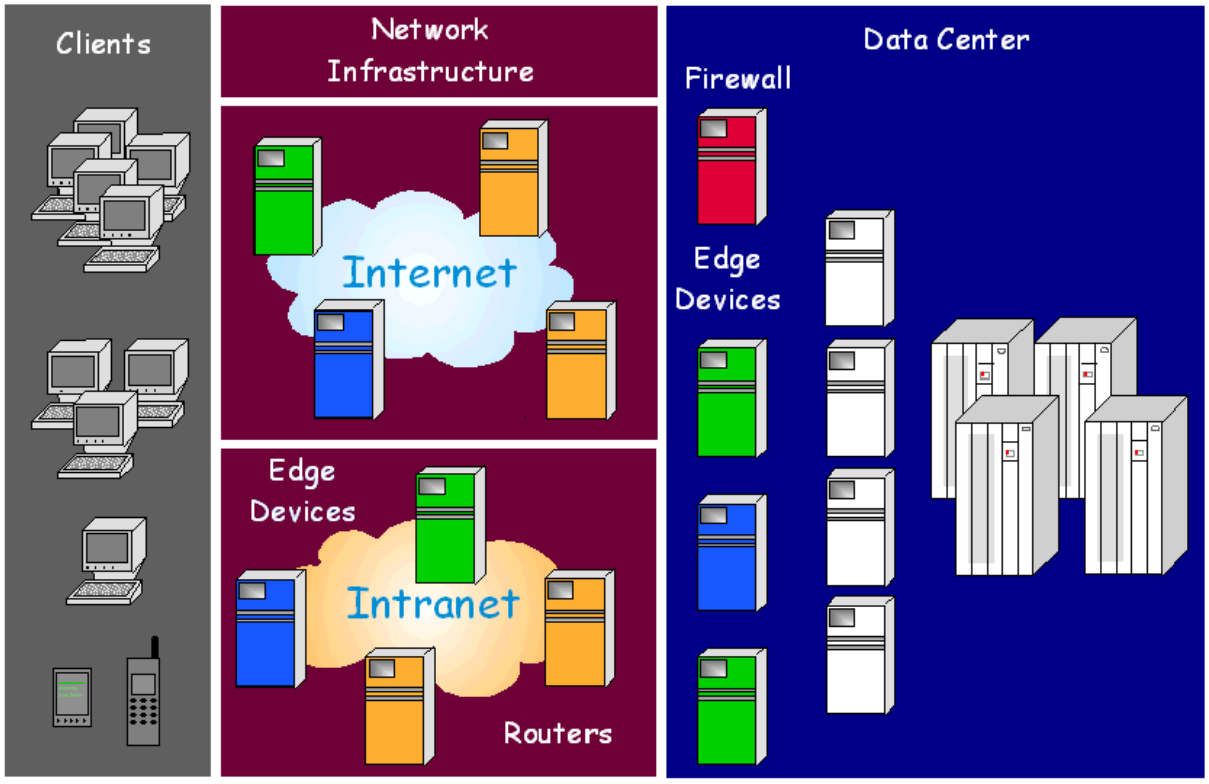

Figure 1: The edges of a network are depicted as white lines between the blocks.

If WTP is installed close to the data source (Data Center), then there is the benefit of ease of deployment and administration because of the typically smaller geographic area it will inhabit. The problem then becomes one of scalability and power. Being deployed closer to the data source (Web servers, Web application servers, databases, etc.) means WTP will have to know how to handle all of

the incoming requests. The numbers and types of these requests could grow exponentially as the reach of the data at the center expands.

If WTP is deployed on the furthest edge of the network away from the data center, then it can be tailored to the capabilities of the clients accessing that particular edge. Scalability is less of an issue since the number and type of requests are more predictable. Plus, transcoding then becomes a natural extension to content distribution frameworks, such as supported through WebSphere Edge Server, where content is generated at the center and distributed to the various edges for caching and transcoding. The drawback to this approach is having to centrally deploy and administer such services. WTP 4.0 not only adds new features to enhance the transcoding function, but also eases the tasks of remote deployment and administration.

Transcoding at the edge

Locating the transcoding process closer to devices offers several advantages, including:

- **Customization**: The process of customizing content by applying stylesheets, annotators or just by mutating it from one markup language to another can be simplified if the scope of device types is known or predictable. There would be fewer stylesheets, annotators, and device profiles to manage. It would also be easier to determine ways to consolidate customizations across device types.
- **Scalability**: On the outer edge of the network, it's easier to react to increases in demand on particular servers when the demand is scaled by the area of coverage. Adding servers and load balancing mechanisms is less disruptive to the rest of the network when isolated to the edges where demands increase. Plus, distant servers on other edges could help take on the load of overloaded network segments.
- **Performance**: If you add caching of transcoded content to the edge through the use of WebSphere Edge Server (described in more detail later), then the path between the content and the device decreases dramatically as popular documents are served more and more out of caches very close to the clients.
- **Stability**: WTP already adds benefit to application deployment by not requiring changes to the application or its serving environment. If WTP is deployed at the edge of the network, there is an additional benefit of data center stability as the innermost networks are not altered to include transcoding services.

There are various ways that WTP can be deployed to enhance the edge of the network:

- **Network Proxy**: As a standard network (forward) proxy, WTP can forward requests from devices or other network components (WAP gateways, authentication proxies) to destination servers and transcode the response on its way back.
- **Reverse Proxy**: Acting as a surrogate to real Web and application servers, the reverse proxy acts as a Web server itself, forwarding requests directed to it on to other servers on the back end, rewriting all URLs in the response to be redirected again through WTP so that all content is transcoded.

- **Application Server filter**: As a MIME filter servlet within WebSphere Application Server, WTP can transcode content after it is generated by a Web application and before it is returned as a response, even as an encrypted response if the request was secure. On the edge, a Web application can help offload server-style processing typically required at the data center.

  Even if WTP is deployed as a MIME filter in the data center, it could be used in conjunction with WTP on the edge to offload and distribute transcoding function. Imagine the filter servlet generating WML from the HTML created by the Web application, or applying a stylesheet to an XML document and allowing WTP on the edge to fragment the resulting document into decks appropriate for the requesting device. The fragments, being closer to the device, will be retrieved much faster than if the request for each fragment had to go all the way back to the data center.

Enhancements in WTP 4.0

As you can see, WTP is already a powerful service for the edge of the network. WTP 4.0 "sharpens the edge" even further with increased integration with other edge services, enhanced administration, and new development tools.

WTP 4.0 introduces support for natively plugging into **WebSphere Edge Server Caching Proxy**, combining content transformation with a best-of-breed caching proxy. This also allows WTP to support HTTP 1.1 and secure connections in the caching proxy's reverse proxy mode as well as the caching of transcoded content. This deployment model also offers a tighter integration with WebSphere EveryPlace Server, where WTP has existed primarily as a stand-alone proxy.

New transforms have been added to the product, including HTML to **VoiceXML**, HTML to **ClipperML** for use with the Palm.Net service, as well as **dynamic translation** of HTML text from one language to another. These services extend the reach of Web data to an even broader audience than before.

Until now, WTP allowed the customization of content based on network and device preferences. New for WTP 4.0 is the ability to also **personalize** the transformation process based on a user's identity. WTP uses a generic Java interface that can be implemented to work with any customer's user profile database. WTP 4.0 offers an initial implementation to work with WebSphere EveryPlace Suite's personalization function.

It has already been described how central administration of edge services can be a challenge. WTP 4.0 offers two new features to greatly improve remote administration. **Request Viewer**, already available as a local debugging tool, now has the ability to connect remotely to an existing, running WTP proxy and monitor requests and responses flowing through the WTP server. **XML Configuration** gives the administrator the ability to represent an entire WTP configuration in a single XML document which can be exported, modified, and imported back into WTP. This makes it very easy to mass distribute server configurations and automate updates without requiring the use of the Administration Console or a central directory.

There have been several enhancements made to XML/XSL processing within WTP, the foremost of which is adding the ability to **select which stylesheet to apply** to the XML document from within the XML document itself. The selection can be based upon new ***wtp-condition*** elements which describe which stylesheet to select and under what condition. There is even support for the ***media*** tag used in stylesheet selection implemented by Cocoon from Apache.

XSL stylesheets and WTP annotators both offer very flexible means of customizing Web data on the edge, especially when the data source is untouchable. Two new tools are introduced in WTP 4.0 to make the development and deployment of stylesheets and annotators easier. The **XSL Stylesheet Editor** aids in the development of stylesheets based on XML documents. The **External Annotation Editor** allows you to build annotation statements to customize HTML pages, which are applied to the HTML at runtime.

Let's go through each of these new enhancements in more detail.

Enhancements in WTP 4.0

WebSphere Edge Server integration

Until WTP 4.0, WTP's integration with WebSphere Edge Server, as well as other caching proxies, has been external. Using what was known as the "enterprise" configuration model, WTP used to be able to store transcoded documents in Edge Server as an external caching proxy, meaning that WTP would check Edge Server for transcoded content before requesting it from the source servers. This still required two proxies: WTP and Edge Server's caching proxy. Additionally, Edge Server supports features that WTP does not, such as HTTP 1.1 and secure socket layer (SSL) connections, that are important for many types of clients. The enterprise model did not solve this issue as WTP would have been specified as the device's proxy, which would mean the device could not exploit HTTP 1.1 features or secure connections.

Now with WTP 4.0, WTP can function as a native plugin to Edge Server's caching proxy. In this mode, Edge Server takes over the function of starting and stopping WTP, as well as handling support of the HTTP protocol (controlling requests and responses) and secure connections. WTP in turn takes advantage of the cache to store and retrieve transcoded content. Two separate proxies are no longer necessary and WTP now takes full advantage of Edge Server's protocol and security features. (Note that this deployment model is in addition to the other models already supported by WTP, namely stand-alone proxy, reverse proxy, and WebSphere Application Server MIME filter servlet.)

WTP inserts itself into Edge Server's caching proxy in two places: after the authentication step and during the transmogrify step. The post-authentication step allows WTP to analyze the request and determine the preferences to use as well as determine if there is any transcoded content already available in the cache. The transmogrify step mutates the resulting document into a device-appropriate

format before being sent back to the requester. During the transmogrify step, WTP will also cache the transcoded document for subsequent requests. See figure 3 for more details.

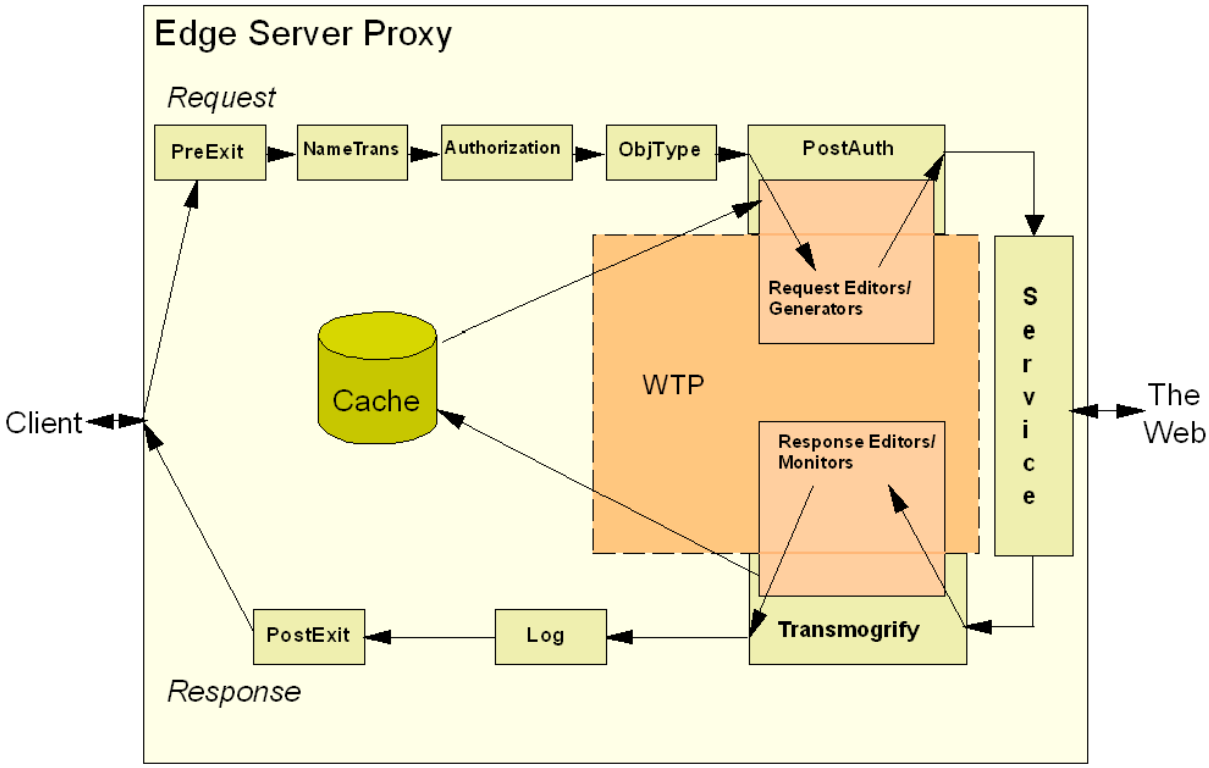

Figure 3: WTP plugin architecture for Edge Server

Secure connections are supported only in reverse proxy mode where Edge Server functions as an SSL endpoint for the connection. Standard network proxies, also known as forward proxies, merely forward secure content to the destination server. There is no capability within a standard proxy to manipulate encrypted data.

In forward proxy mode, Edge Server with WTP installed has the outward appearance of behaving no differently than WTP's independent proxy, except that Edge Server is responsible for processing the request and generating the response instead of WTP, and Edge Server's caching capabilities will substantially improve response times. In reverse proxy mode, however, the two proxies do behave differently. Edge Server by itself, when configured as a reverse proxy, simply maps requested URLs to specific servers on the back side. Any page that is returned which contains links to other servers will bypass Edge Server. When running as an independent reverse proxy or as a plugin to Edge Server running as a reverse proxy, WTP parses the returned document and rewrites all links in that document to be forwarded back through WTP (and thus Edge Server) to ensure that all content is transcoded, not just the original request. This means Edge Server's reverse proxy behaves more like a portal to all Web content when WTP is plugged in instead of merely a router for specific URL patterns.

A new feature has been added to WTP 4.0 in support of the Edge Server integration model for allowing transcoding to be turned off on a per-device basis. This allows for devices such as desktop browsers or

function-rich HTML browsers on PDAs to use Edge Server as a caching proxy without incurring the overhead of any (even minimal) WTP processing. To turn off transcoding for a particular device, go to that device's preference profile in WTP's Administration Console and deselect the **Disable transcoding** preference. By default, the preference is available for desktop browser profiles only. You can use the Profile Builder application to expose the preference for other devices as well.

New transcoders

HTML to VoiceXML

The VoiceXML transcoder takes an HTML page, written in terms of a two-dimensional visual display, and builds a one-dimensional, or serialized, voice interface. That interface, defined using VoiceXML, is then read by a voice browser, such as WebSphere Voice Server, and converted to speech.

There are several challenges associated with converting a visual representation to an audible one. The foremost is navigation. In a visual page, the eye can randomly navigate a page. Information is fed to the ear sequentially, not randomly, so the listener must be able to quickly and logically navigate through content without having to listen to all of it. The VoiceXML transcoder organizes content by topic and type, giving the listener menus as fast paths to main content and links. Other issues involve speech synthesis and recognition quality, listener attention span and ability to process data and remember options. These issues make the voice interface's usability a top priority.

It is not likely that a voice application will be based totally on transcoded HTML content. The application developer will want tighter control over navigation and content. Instead, the VoiceXML transcoder should be seen as a natural extension to the voice application, adding the ability to fetch dynamic data from external Web sites and incorporate that data within the voice application. Refer to the white paper "Guidelines for a VoiceXML Solution using WebSphere Transcoding Publisher" (found on http://www.ibm.com/software/webservers/transcoding/library.html) for more information.

HTML to ClipperML (for Palm.Net)

The Palm Transcoder transforms HTML to ClipperML, a markup used by the Palm.Net service and is compatible with the Palm Mobile Internet Kit and Palm OS 3.5. The Palm.Net service compresses Web pages and sends them to the Palm device for browsing using the Mobile Internet Kit. To be compressed, the page must meet specific HTML authoring guidelines, or limitations, which means that certain HTML elements are not supported.

The Palm.Net service cannot be configured with other proxies, so WTP must be configured as a reverse proxy or as a MIME filter in WebSphere Application Server so that the Palm device can direct requests to WTP through Palm.Net. The Palm Transcoder will convert HTML pages to web clippings by removing the unsupported elements, then forward the result back through Palm.Net to the device.

Machine Translation transcoder

“Machine translation” refers to the ability to dynamically translate text from one language to another. WebSphere Translation Server offers that function, and WTP 4.0 provides a transcoder that interfaces with one or more Translation Servers to dynamically translate Web content.

WTP can be configured to use many Translation Servers at once and it will automatically balance translation requests across them, so as not to overburden any one server. WTP will also dynamically learn the translation capabilities of each server since the language combinations are configurable (English to French, Spanish to English, English to Japanese, for example) and will send translation requests to the appropriate Translation Server based on the desired target language and the language of the source. For instance, if a browser indicates that its preferred language is Italian and the HTML source is in English, then WTP will forward the HTML document to a Translation Server which supports the English-to-Italian combination. If no appropriate combination is found, the document is forwarded untranslated.

As you can imagine, translating a document can be a difficult chore. The accuracy of the translation depends highly on the ability to determine the subject. WTP provides annotation language extensions for giving subject hints within HTML documents as they are processed. For example, if you know the subject matter of all pages you retrieve will involve computers, the Internet, eCommerce, etc., you would use WTP annotators to add each of these keywords as subjects to the HTML so Translation Server will do a better job selecting an appropriate translation. Annotation can also be used to simplify the page, removing content that isn’t needed by the device, so the process of translation will be faster.

User-based preferences

Before version 4.0, WTP based the decision of how to transcode source on network and device preferences. New to WTP 4.0 is the addition of user preferences into this decision making process, also known as preference aggregation.

Network preferences allow WTP to customize transcoding based on the network port used by the device when connecting to WTP. Wireless customers may be configured to use a different port than wired, or LAN, customers and thus have preferences set to minimize the amount of data sent to the client to save bandwidth, such as turning images into links, or removing images all together. Device preferences tailor the transcoding process based on the capabilities of the device making the request. For example, WAP phones would have GIF and JPEG images turned into monochrome bitmaps and HTML pages converted into WML. All of these preferences are objective decisions made based on facts learned from the request. User preferences allow the device user to override network and device preferences, when allowed, with personal preferences for how the content should appear on the device.

Examples of typical user preferences for transcoding would be whether to convert images to links, remove images, or what language and subject to use when translating documents from one language to another. Any editable preference represented in device and network preference profiles can be used in

a user profile. There are certain, non-editable preferences which are assumed to be facts about devices that are not subject to override by a user, such as device screen size and document size.

WTP makes no assumption about how user profiles are stored, other than that they are accessible using Java APIs and that the user's identity can somehow be extracted from the request. WTP therefore provides Java interface definitions to access user profiles and user identities. If the user's identity is provided within the client request as an HTTP header field, then the header field can be specified in WTP's configuration as the means of extracting the user's identity. Otherwise, customers will have to implement the appropriate Java interface to provide that information.

WTP is only a consumer of user profile information and does not support any form of administration for user data. It is assumed that WTP will be integrated with some existing user profile management system with its own administration. However, WTP 4.0 does provide support for WebSphere EveryPlace Server's user profile management scheme. Implementations of the Java interfaces are provided for accessing user data from a central directory using the Lightweight Directory Access Protocol (LDAP) and the user's identity is extracted from the client request using predefined HTTP header fields. Again, these implementations and header fields are for interoperability with WebSphere EveryPlace Server and can be replaced with similar implementations to match other user profile management systems.

If a request is made to WTP with a user identification included, and implementations to the Java interfaces are provided, and the user profile can be successfully located, then the user's preferences will be included during preference aggregation. Otherwise, preference aggregation will use the default user preferences supplied within WTP. Successfully retrieved user profile data is cached for 24 hours.

Request Viewer enhancements

Before WTP 4.0, the Request Viewer tool was simply a running instance of the WTP proxy with a window that allowed you to watch the requests and responses flow through WTP and see the inputs and outputs of each transcoder. It required that the Request Viewer instance of the proxy be used in analyzing situations instead of the original server where the situation was observed. In WTP 4.0, the Request Viewer can now be attached to running instances of the WTP proxy, running either locally or remotely. This allows the administrator to centrally monitor the health of WTP proxies running anywhere in the network, even out on the edge.

By default, WTP will not allow a Request Viewer to attach from a remote location unless the Request Viewer's machine has been added to an authorization list. Using the Administration Console on the WTP server, the administrator adds to the authorization list the TCP/IP host name or address of the remote Request Viewer machine, indicating that Request Viewer may connect to the local WTP server. Note that if Request Viewer is run on the same server as WTP, no authorization is necessary. Only one Request Viewer at a time is allowed to connect to any given WTP proxy server.

Request Viewer currently supports WTP running as an independent proxy or reverse proxy. Support for WTP running as a plugin to WebSphere Edge Server or as a MIME filter servlet in WebSphere Application Server will be added in a later release.

XML Configuration

XML Configuration enables the administrator to export all of WTP's configuration to a single XML document. This document can be edited to add, subtract, or modify resource definitions, then imported. This makes it very simple to register a large number of similar resources, such as XSL stylesheets and annotators, and even mass-distribute configurations to many remote servers. Updates to WTP could be automated as well, using scripts to modify the XML configuration file and import the results. The changes take effect immediately. The administrator is no longer required to use the Administration Console to make updates to WTP.

XML Configuration is also used when migrating from previous versions of WTP to version 4.0. Product installation detects that an older version of WTP is already installed on the system, uses a special version of XML Configuration to export the older configuration, installs WTP 4.0, then imports the original configuration. The import function recognizes that the data is from a prior release and migrates it to the current release as appropriate.

XML Configuration can also be used to backup a WTP configuration, such as part of a periodic system backup or just prior to a major configuration change.

XML/XSL enhancements

WTP 4.0 adds support for the <?xml-stylesheet?> XML tag, which specifies what stylesheet to use when processing the XML document. This allows the XML document author to include a reference (by URL) to the stylesheet that should be applied to the document by WTP and obviates the requirement that the stylesheet be registered in WTP. The administration savings of this approach are large, especially if WTP is deployed at the edge of the network, far from the data center where the XML documents are generated.

In fact, the XML document could be generated to include a reference to a stylesheet which creates an HTML representation of the XML, which is then transcoded to a more device-appropriate markup language, like WML. This makes use of HTML as an intermediate format and is a way to reduce the number of stylesheets required to support various markup language versions of a single XML document. Refer to the **IBM developerWorks** article "Spinning your XML for screens of all sizes" (http://www.ibm.com/developerworks/library/i-multx/) for more information.

The fact is, though, that an XML document owner may want stylesheets to handle transforming XML to a specific markup language, in order to have better control over the appearance and layout. A WTP administrator can accomplish this by registering in WTP several stylesheets for the same XML

document which are applied based on device type or other request characteristics. The XML document owner can also do this in one of two other ways. The first is to adopt the use of an additional attribute on the <?xml-stylesheet?> tag called *media*, used by the Cocoon Project supported by the Apache Software Foundation. The second is to use the more flexible <?wtp-condition?> tag developed specifically for WTP.

Cocoon compatibility

Cocoon introduced the media attribute to the <?xml-stylesheet?> tag as a way to map a stylesheet to a specific device. The mapping is provided in a flat file and the devices are identified by their User-Agent HTTP header value. Using the *media* attribute as a differentiator, multiple <?xml-stylesheet?> tags can exist within the same XML document and the correct stylesheet is selected based on what device (media) requests the document. Here is a sample of several <?xml-stylesheet?> tags within an XML document which use the *media* attribute:

```
<?xml-stylesheet href="hello.xsl" type="text/xsl" media="wap"?>
<?xml-stylesheet href="hello-text.xsl" type="text/xsl" media="lynx"?>
```

And here is a sample media mapping table:

```
browser.0=explorer=MSIE
browser.1=opera=Opera
browser.2=lynx=Lynx
browser.3=java=Java
browser.4=wap=Nokia-WAP-Toolkit
browser.5=netscape=Mozilla
```

In the above example, the hello.xsl stylesheet will be selected if the requesting device has a User-Agent header value of “Nokia-WAP-Toolkit”. Likewise, the hello-text.xsl stylesheet will be selected if the requesting device has a User-Agent header value of “Lynx”.

WTP supports the media tag as well as the mapping table. An ImportCocoon script is provided to import the media mapping table into WTP’s configuration. You are limited, however, to selecting stylesheets based solely on a device’s User-Agent value. It is recommended that this support in WTP be used as a migration step to the more flexible <?wtp-condition?> tag.

More flexible stylesheet selection

The <?wtp-condition?> tag offers the same type of functionality as the <?xml-stylesheet?> tag plus the media attribute but it allows you to select the stylesheet based on any preference profile in WTP as well as any HTTP header value. Preference profiles are referenced by their profile name and HTTP headers by their actual header name, all within the *condition* attribute, like so:

```
<?wtp-condition  stylesheet=http://xslserver/wtp/test-wml.xsl
condition="(device=WML-Device) &  (url=*test.xml)"?>
<?wtp-condition  stylesheet=http://xslserver/wtp/test-palm.xsl
condition="(device=Palm-Neomar) &  (url=*test.xml)"?>
```

The test-wml.xsl stylesheet will be selected and applied to the XML document by WTP when a device matching the WML-Device preference profile requests an XML document called test.xml. Likewise, the test-palm.xsl stylesheet will be selected when the same XML document is requested by a device matching the Palm-Neomar preference profile. The <?wtp-condition?> combines the flexibility of selecting the correct stylesheet based on attributes of the request with the ease of administration by containing the selection logic within the XML document itself and not requiring administrators to register those stylesheets on every WTP server.

Note that WTP will select the first matching <?xml-stylesheet?> or <?wtp-condition?> tag that it encounters, not necessarily the most accurate match. Also note that stylesheets registered in WTP which also match a request will override <?xml-stylesheet?> and <?wtp-condition?> tags, unless the **checkForDocumentStylesheetsFirst** preference is set to *true* in the XMLHandler transcoder property file.

External Annotation Editor

What is annotation?
Annotation refers to the ability to customize the content of an HTML page using an XML-based language. Annotation allows you to very quickly identify areas of an HTML page to be removed, kept, or exchanged with other content. An alternative to using annotation is to write a Java clipper to manipulate the page directly, but this requires Java programming skills and intimate knowledge of the page's layout. Annotation is easy to write, can be just as effective with only a general knowledge of a page's layout, and best of all, requires no Java programming.

There are two types of annotation: internal and external. Annotation instructions can be inserted directly into the HTML source as comments, called internal annotation. This works best if you have access to the HTML source. External annotation refers to annotation instructions that are maintained in a separate file, called an annotator, which is then applied to the HTML page by WTP when requested. This works best if you do not have access to the HTML source or if a single annotator could work for several different HTML pages. External annotators are also best suited for WTP deployments at the edge of the network where custom annotators can be built to suit the needs of the devices being served on a specific edge. Also consider the fact that functions deployed on the edge of networks are often done so because access to the data center is impossible or restricted. Internal annotation may not be possible.

WebSphere Studio Version 3.5.2 introduced support for inserting internal annotation instructions into HTML documents. Until WTP 4.0, there was no tool for generating (external) annotators.

The new External Annotation Editor
The annotation language is relatively easy to use, but since annotation instructions are based on the location of elements in the original document, it is easy to formulate annotations incorrectly. The External Annotation Editor alleviates this problem by allowing you to load the HTML document that needs to be annotated. The editor parses the document and gives you two views on the left-hand side of the editor: a tree, or document object model (DOM) view, and a source view. You simply select the desired

elements in the page and apply annotation instructions to it. You see the annotator being built on the right-hand side as you go. See figure 4 for details.

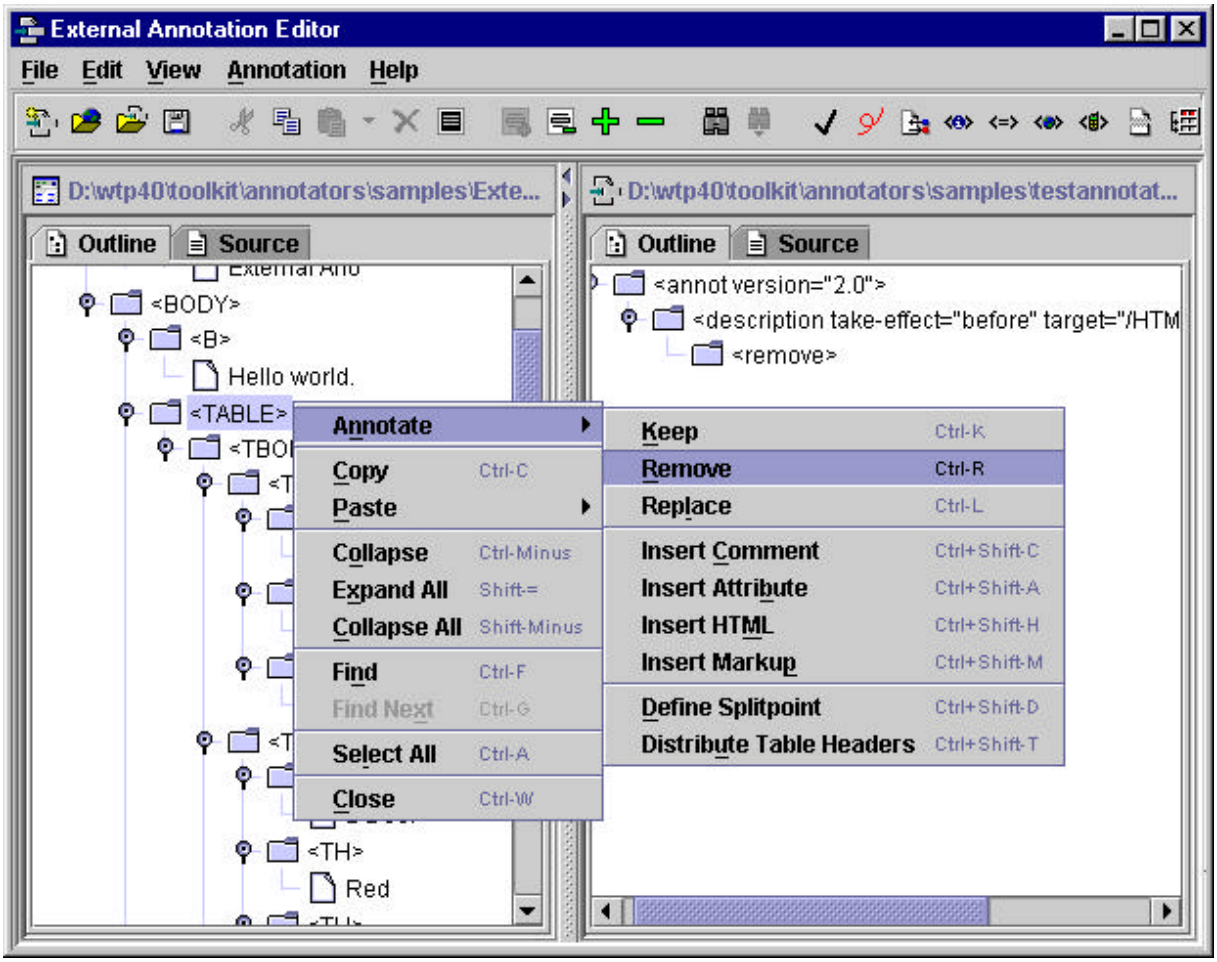

Figure 4: External Annotation Editor

As you can see by the figure, the editor allows you to easily select areas of the page to keep, remove, replace, as well as for inserting new content. You can insert an HTML comment, additional raw HTML, additional attributes, or additional markup of some variety, such as JavaScript.

The annotation language is also used in conjunction with other WTP functions to facilitate custom transcoding actions. Working with the fragmentation engine, which splits large pages in to smaller fragments that can be consumed by small devices, the editor can be used to specify a fragment breakpoint hint, or “suggestion” to WTP of where a good place for a fragment to begin. You can also specify that table headers be distributed to each cell of each row in case the table is converted to a list, to improve readability.

As you can see, the External Annotation Editor provides an easy to use mechanism for generating annotators, using features you are already familiar with, such as HTML source and tree views, drag and drop, and wizard-based task guides.

XSL Stylesheet Editor

XML is a representation of data. To render that data in some displayable format, such as HTML or WML, or even to mutate it into some other data format, you typically apply a stylesheet. A stylesheet is a collection of rules that define conditions to look for in an XML document and what to do when those conditions are met. A rule contains a condition and an output. When a condition is met, such as a certain element in the document is found, the output might be how to display that element, such as what HTML tags to use, whether to italicize the output, etc.

WTP has always had the ability to apply appropriate stylesheets to XML documents, but the creation of the stylesheets is the responsibility of the customer since the stylesheets themselves depend on the nature of their corresponding XML documents. Creating a stylesheet can be a laborious task, requiring in depth knowledge not only of the XML source document, but also of the extensive and flexible XSLT language.

XSLT, the language in which XSL stylesheets are written, is relatively new. There is a shortage of development tools in the market to aid the development of stylesheets--that is, until now. WTP 4.0 introduces the XSL Stylesheet Editor, a tool which helps you build a stylesheet for rendering XML data in some other format.

The XSL Stylesheet Editor organizes XML documents and their stylesheets into projects. When an XML document is loaded into the project, it is parsed into a document object model (DOM), which is then used to render the tree view of the XML document. A source view is also provided. By default, a rudimentary XHTML rendering of the data is given on the right-hand side, called the design view, from which modifications can be made to get the desired rendering. Text and tree views of the rendered results as well as the stylesheet are also available. Text source views are read-only, while the tree view allows for manipulation of the data. See figure 5 for details.

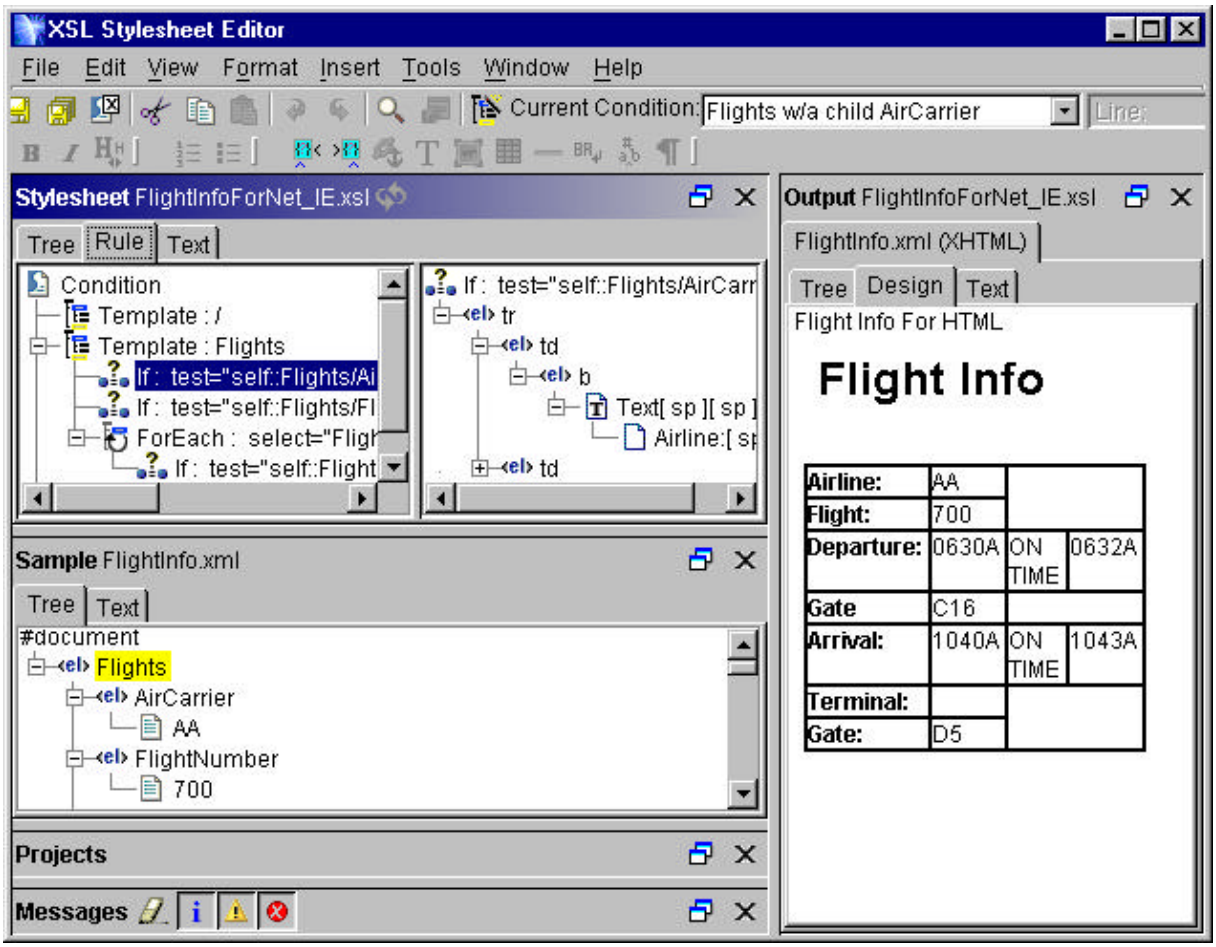

Figure 5: XSL Stylesheet Editor

There are three steps to building a stylesheet. First you select the element you want to work with, either in the XML source or in the output design view. Then you specify the condition under which you want this element selected, such as all matching elements, or only those elements that are children of other specific elements, etc. Finally, you specify what to do with the elements when the condition is met, such as special formatting for the element's data. All stylesheet creation and editing tasks revolve around these three steps.

The stylesheet editor is also useful for testing the effect of applying existing stylesheets to XML documents, then altering the results to modify the stylesheet. Figure 5, for example, shows the FlightInfoForNet_IE.xsl sample stylesheet being applied to the FlightInfo.xml sample XML file, both of which are shipped with WTP.

Conclusion

WebSphere Transcoding Publisher Version 4.0 adds strength and flexibility to an already sound transformation platform. Transcoding can now be based not only on what network you are using and what device you use, but who you are. The External Annotation Editor and XSL Stylesheet Editor,

together with the XML enhancements, user-based personalization, and annotation language improvements, provide the means to customize content far beyond simple translation of one format to another.

As enterprises open their borders to Internet users, their data centers can easily be overwhelmed with maintaining knowledge of not only the data, but who is accessing it and how. Relocating the function that tailors content for the device and user closer to the device and user not only relieves the servers in the data center of that burden, but also narrows the function's scope, building a system which is easier to administer and scale.

WTP 4.0 has made several improvements which help deploy transcoding function to the various edges of the network. Indeed, all deployment models of Transcoding Publisher have benefited from these enhancements. The new transcoders, user personalization, XML Configuration, performance and usability improvements all contribute as much to WTP running within WebSphere Application Server as within WebSphere Edge Server. In fact, it may be more correct to say that WebSphere Transcoding Publisher Version 4.0 sharpens the edge not only of your network, but of your enterprise as well.

Trademarks

IBM, WebSphere, and Everyplace are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.