Lotus Developer Domain
The Technical Resource for Lotus software
LDD Today

Level: Beginner
Works with: Domino 6
Updated: 01-Oct-2002

by
John
Chamberlain

A well-designed Web site allows visitors to easily get around and find the information they want without becoming impatient or frustrated. This is often achieved by having plenty of links and maps to help visitors navigate. One of the big problems when maintaining a site is keeping all the links and maps up-to-date. Nothing discourages a visitor faster than encountering broken links.

Since the organization of a Web site can be quite complex and changeable, it's helpful to have a way for URLs and links to remain valid even when the underlying resources are rearranged. This not only helps with maintenance, it also benefits your regular customers who bookmark interesting places on your site.

In the *LDD Today* article, "**Building Web applications in Domino 6: A tutorial on Web site addressing**," we discussed the basics of setting up Web sites in Domino 6. In this article, we will explore how to use Web site rules to help maintain the organization of your Web site. Web site rules provide this help by allowing you to control the interpretation of incoming URLs. Rules have two main uses:

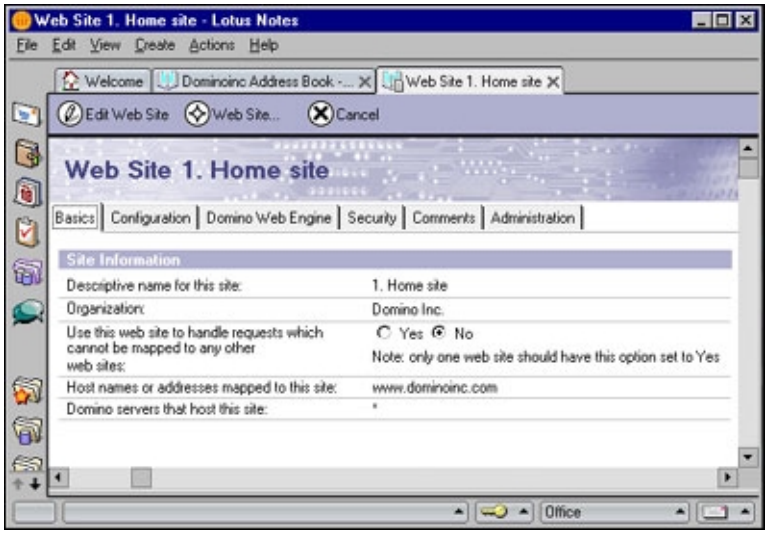
- To allow you to create a user-friendly and consistent navigation scheme for the site that is independent of the site's actual physical organization.
- To allow parts of the site to be relocated or reorganized without breaking existing links or browser bookmarks.

This article will be of interest to all system administrators and webmasters who plan to host Web sites with Domino 6. It assumes a familiarity with the Domino Directory and administrative tasks.

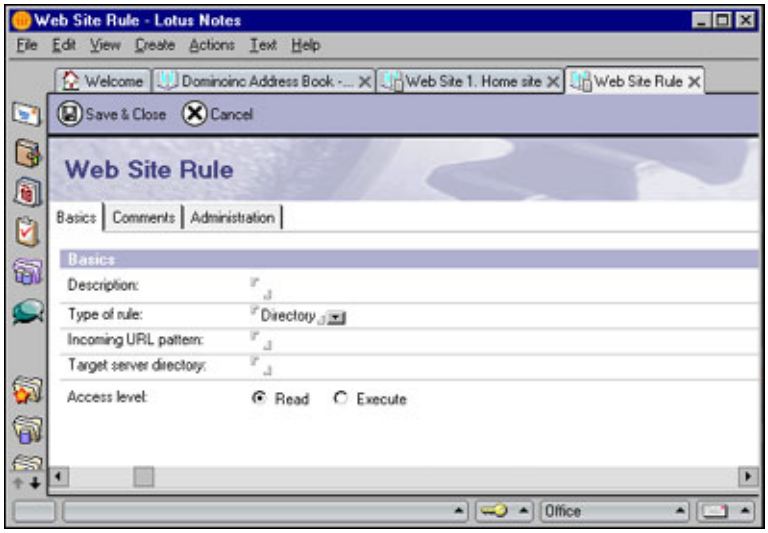
The Web Site Rule document

Web site rules are defined in the Domino Directory, by creating response documents to Web Site documents in the Server\Internet Sites view. As with all other information defined for Web sites, a rule applies only to its parent site. However, because rules are implemented as simple response documents, if you want a rule to apply to more than one site, you can just copy and paste the rule from one site to another.

Web Site Rule response documents are created from an action button in the Web Site document. Here's the Web Site document for www.dominoinc.com, which we created in the *LDD Today* article, "**Building Web applications in Domino 6: A tutorial on Web site addressing**:"

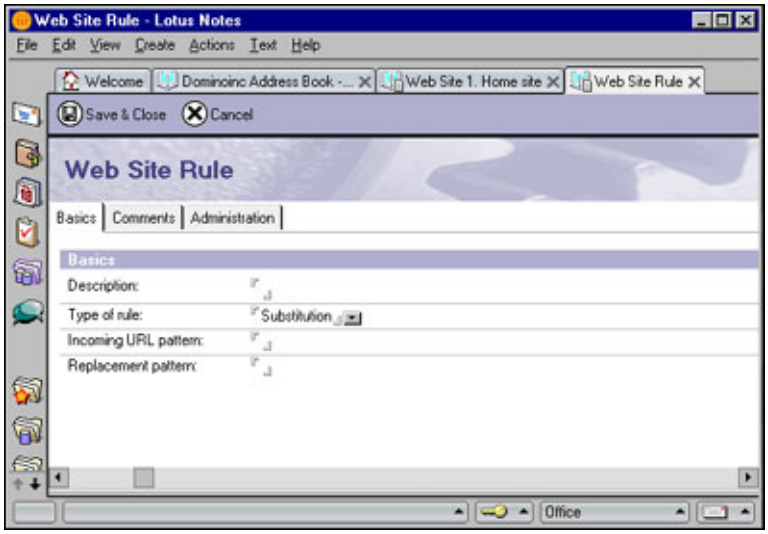


To create a rule for this site, click the Web Site action button and choose Create Rule. This will bring up an empty Web Site Rule document:



The Type of rule field in this form allows you to choose from the three available types of rules: directory, redirection, and substitution. In this article, we'll discuss redirection and substitution rules. Look for information about directory rules in a later article, which will discuss how to set up Web site file-system resources.

When you select Substitution for the Type of rule field, the form changes to display the fields for substitution rules:



The Description field can hold any descriptive text you want to use to identify or explain the rule.

The Incoming URL pattern field contains a "template" that is matched against URLs sent from browsers or other HTTP clients. The pattern can include the special asterisk wildcard character (*), which matches any number of incoming characters (that is, zero characters or more). For example, the pattern /shop/* will match any URL that starts with /shop/.

The Replacement pattern field contains a template that describes how the server should modify a matching URL. The part of the URL that matches the incoming pattern is replaced by the replacement pattern. The replacement pattern can also include wildcard characters, as we'll discuss in detail in the **How paths are matched by rules** section later in this article. For example, if the rule for /shop/* has the replacement pattern /market/*, then the rule will turn the URL /shop/pets into /market/pets.

If you set the rule type to Redirection instead of Substitution, the Replacement pattern field is replaced by a field labeled Redirect to this URL. This is also a template that may contain wildcards.

Before you can understand the use of the incoming and replacement patterns, you need to understand how the Domino 6 Web server handles URLs sent from a browser client.

URLs in a hostile world

As Web administrators know all too well, the Internet is constantly under attack by malicious persons ranging from novice script-kiddies to sophisticated cyberterrorists. As soon as a vulnerability is discovered in a piece of Internet software, attacks are crafted to take advantage of that vulnerability and are launched—sometimes against specific targets, but often against the whole Internet population. A public Web server is often hammered by attacks that are intended for other targets. Sometimes an attack that is launched against a particular piece of software will have serious effects on other vendors' software.

To defend against this hostile environment, the HTTP task of the Domino 6 Web server implements an extensive array of URL filtering and validation procedures. The intent of these procedures is to reduce every incoming URL to a safe, normalized form *before* the URL is passed to an application for processing. If the HTTP task cannot normalize a URL, the request is rejected immediately, and an error (generally containing a short, nonspecific message) is returned to the sender.

When the Domino 6 Web server receives a URL, the HTTP task performs the following procedures to normalize the path.

1. Check the overall length

The overall length of the URL is checked against the Maximum URL value specified in the Server document (located under Internet protocols on the HTTP tab). This allows the server to immediately reject requests from attackers who send extremely long URLs to probe for buffer-overflow vulnerabilities.

2. Parse the URL
The URL is parsed into its component parts. The general syntax of an HTTP URL is:

http://<host>/<path>?<query>

For example, if you enter the URL:

http://www.dominoinc.com/products.nsf/index?OpenPage

in a browser, www.dominoinc.com is the host, /products.nsf is the path (the beginning slash is considered part of the path), and OpenPage is the query (Domino makes use of the query string to pass database commands such as OpenPage, OpenDocument, and so on).

3. Decode hex-encoded sequences
All hex-encoded sequences in the URL path are decoded and replaced with their character equivalents. For example, /Smith%26Jones/welcome becomes /Smith&Jones/welcome. Domino 6 supports traditional hex-encoding as well as UTF-8 encoding for Unicode. Attackers often try to evade server filters by multiply encoding special characters. For example, a backslash may be singly encoded as %5c; those characters can themselves be encoded as %25%35%43. To thwart these attacks, the Domino 6 HTTP task iteratively decodes the URL path until no more decoding can be done. Also, if the decoder detects an invalid hex or UTF-8 sequence, the request is immediately rejected. The CodeRed and Nimda worms make use of "overlong" UTF-8 sequences such as %c0%af to represent a forward slash; Domino 6 will reject any URL containing overlong sequences.

4. Resolve all directory-navigation sequences
All directory-navigation sequences in the URL path are resolved. Both forward slashes and backslashes are recognized as path delimiters, and multiple slash characters are ignored. For example, /abc/..\\xyz/.//123 becomes /xyz/123. Because the Domino 6 HTTP task does this "de-navigation" up front, it is impossible for an attacker to back up out of the Domino HTTP root directories. The default root directories on Windows are c:\Lotus\Domino\Data for nsf requests and the c:\Lotus\Domino\Data\domino subdirectories for file-system requests.

For example, the Nimda worm tries to run cmd.exe by sending the URL path "/scripts/..%255c..%255cwinnt/system32/cmd.exe" to a server. Domino 6 decodes and de-navigates this path to: /winnt/system32/cmd.exe. Since this appears to be a file-system request, Domino applies this path to the HTML root directory and actually looks for the file c:\Lotus\Domino\Data\domino\html\winnt\system32\cmd.exe, which of course is not going to exist. Thus the HTTP task will return a Error 404 - File Not Found response to this attack.

5. Check the number of segments
The number of segments in the URL path is checked against the "Maximum number of URL Path Segments" value specified in the Server document (which is also located under Internet protocols on the HTTP tab). This prevents attacks that try to overwhelm the server's file-system code with a huge number of path segments (for example, /a/b/c/d/e/f/g/h/i/j/k...).

How paths are matched by rules

After an incoming URL has been normalized, the HTTP task scans the rules defined for the Web site to determine if the URL is to be modified in any way. Only the URL path is used for pattern matching; the query string is saved away for the application's use. Therefore, the patterns you specify for a rule's Incoming URL pattern field should never include a host name or query string. Because matching is done on normalized paths, you do not need to worry about trying to match all possible variations of a raw URL. For example, if you want to match the welcome page for Smith&Jones, you can just specify one rule for /Smith&Jones/welcome; you don't need to create additional rules for /Smith%26Jones/welcome, /Smith%26Jones/./welcome, and so on.

As mentioned above, patterns can contain the asterisk wildcard character (*) to match zero or more characters. For example, if you specify the pattern /petstore/*dogs, the minimum matching URL path is /petstore/dogs, and it will match any other path that starts with /petstore/ and ends in dogs, such as /petstore/aisle/bigdogs.

Also, pattern-matching is not case-sensitive. The pattern /petstore/*.nsf will match both /petstore/specials.nsf and /PetStore/Specials.NSF.

The replacement or redirection pattern can also contain wildcards. These wildcards are replaced by the parts of the path that matched the corresponding wildcards in the incoming pattern. For example, the following substitution rule:

Description: Product specials database

Type of rule:	Substitution
Incoming URL pattern:	/*/specials/*
Replacement pattern:	/*/specials.nsf/by*?OpenView

will transform the path /petstore/specials/month into /petstore/specials.nsf/bymonth?OpenView.

You'll notice that the replacement pattern contains the query string ?OpenView. We mentioned above that query strings aren't allowed in incoming patterns, but they are allowed in replacement patterns. If the new URL is given a query string by the replacement pattern, it will override any query string that was in the original URL.

Substitution rules take priority over redirection rules (and redirection rules take priority over directory rules). The substitution and redirection rules for a Web site are kept in separate tables. Within each table, the rules are ordered by the length of the incoming URL pattern, from longest pattern to shortest. This insures that more specific rules will get matched before less-specific ones. For example, suppose you create the following two rules:

Description:	General mall pages
Type of rule:	Substitution
Incoming URL pattern:	/mall/*
Replacement pattern:	/mall.nsf/*?OpenPage

Description:	Each store has its own nsf
Type of rule:	Substitution
Incoming URL pattern:	/mall/*/*
Replacement pattern:	/*.nsf/*?OpenPage

The URL path /mall/pets/welcome will match the longer pattern and be transformed to /pets.nsf/welcome?OpenPage, whereas the path /mall/welcome will match the shorter pattern and be transformed to /mall.nsf/welcome?OpenPage.

After normalizing the URL path, the HTTP task first searches the substitution rule table. If a matching rule is found, the task generates a new URL path from the replacement pattern and re-normalizes the path.

Next, the HTTP task will try to match the path against the redirection rule table. As explained in the **Redirection rules** section below, there are two types of redirections: external, which are sent back to the browser, and internal, which are processed within the server. If the path matches an *external* redirection rule, the HTTP task generates a new URL based on the redirection pattern and immediately returns that URL to the browser. If the path matches an *internal* redirection rule, the HTTP task generates a new path, normalizes the path, and then searches the redirection rule table again. Because the HTTP task does this recursive search through the redirection rule table, you can write broad-based redirection rules that will capture URLs no matter what substitutions or redirections have already been applied. Because having a recursive search means that there is the potential for getting into an infinite loop if you mistakenly write redirection rules that match each other, the HTTP task has a built-in recursion limit of ten.

A word about security

We've pretty well covered the basics of Web site rules. Now we're ready to delve into the details of substitution and redirection. But first, an extremely important word about security:

Do not use rules to enforce security!

Because rules modify incoming URL paths, you may be tempted to try to use rules as part of your site security. Don't do it!

Rules are a management feature for the webmaster. Rules are *not* a security feature! If you use a rule to try to hide part of your Web site you are guilty of "security of obfuscation," which has been proven time and time again

to be no security at all. In fact, it may be worse than no security at all because it gives you the illusion of security, very likely causing you to become careless in other areas.

Let me give you an example. Suppose your Domino 6 Web server is hosting sites both for your internal users and for the general public. All of your internal databases are in the subdirectory data\internal, and all of your public databases are in data\public. You have two Web sites defined: www.myco.com for public users and internal.myco.com for your internal users. Internal.myco.com only allows SSL access with client certificates, so it is a very secure site. Because you know that no unauthorized persons can access internal.myco.com, you figure you can save yourself some administrative headaches by setting the database ACLs for everything in data\internal to allow Anonymous access. To prevent public users from accessing these databases, you create a redirection rule for www.myco.com that redirects all URLs specifying /internal/* to an error page. Your internal databases are safe, right?

Wrong! You have committed a terrible security blunder. First of all, you forgot that a database can be accessed from the Web by its replica ID. For example, internal\names.nsf might be accessed by http://www.myco.com/85256ab40043144e?OpenDatabase. Although replica ID are strings of 16 hex digits, they are not truly random and with today's cheap computing power and high bandwidth it is trivial for an attacker to find valid replica IDs by storming the server with requests. Second, you may defeat the security yourself by moving or renaming the database. If you move internal\specs.nsf to planning\specs.nsf, you are likely to remember to fix up the links and rules for internal.myco.com but forget to fix the redirection rule under www.myco.com. Now anybody can open the database just by requesting http://www.myco.com/planning/specs.nsf.

The lesson to take away from this example is simple: If you need to protect a resource on your site, you *must* protect the resource with security *on the resource itself*. You must not try to "hide" the resource with rules, hard-to-guess names, or other obfuscation. You should only use the true security features provided by Domino, such as ACLs for databases and documents.

By the way, this lesson is true for all Web servers, not just Domino. Any Web server configuration feature that relies on obfuscation is not secure. With that understood, let's get into details of the different types of rules.

Substitution rules

The purpose of a substitution rule is to replace one or more parts of an incoming URL path with new strings. The two main uses for substitution rules are:

- You need to reorganize your site and do not want to have to rewrite all the links within the site.
- You want to provide user-friendly aliases for complex URLs.

Let's look at some common scenarios where substitution rules are useful.

Example 1: Moving files within a site

Suppose you need to move your Domino databases from the "shopping" subdirectory to the "market" subdirectory. Pages on your site contain many links that reference the shopping directory, for example:

```
<a href="/shopping/products.nsf">Browse our products</a>
```

Of course, after you move products.nsf to the new directory, this link won't work. To fix it, you could manually update the link to:

```
<a href="/market/products.nsf">Browse our products</a>
```

But that could be very time-consuming and error-prone if you have a lot of broken links to fix. A much easier way to solve the problem is to create a substitution rule that maps the old directory to the new directory:

Description:	Site databases moved from "shopping" to "market"
Type of rule:	Substitution
Incoming URL pattern:	/shopping/*
Replacement pattern:	/market/*

With this rule in place, when the user clicks the link:

Browse our products

the server transforms the URL path to:

/market/products.nsf.

Example 2: Converting a non-Domino site to Domino
Suppose you have a simple shopping-mall Web site whose content is HTML files organized into directories, one directory for each store in the mall. You are upgrading this rudimentary site to Domino and have created a Domino database for each store, with the HTML files copied into Domino pages. As with the previous example, this will break the static links in the pages. But if you make the database names the same as the directory names, and the page names the same as the file names, you can create a single substitution rule that covers the entire site:

Description:	Convert "stores" files to Domino pages
Type of rule:	Substitution
Incoming URL pattern:	/stores/*/*.html
Replacement pattern:	/stores/*.nsf/*?OpenPage

This rule illustrates the use of multiple wildcards. The first wildcard transforms the directory name into a database name and the second wildcard transforms the file name into a Domino page name. For example, if the user clicks on the link:

Petstore

the URL path will be transformed to:

/stores/petstore.nsf/welcome?OpenPage.

Example 3: Creating an alias for a complex URL
You can also use substitution rules to provide a simple bookmark for a very complex URL. The following rule creates an alias for a search request:

Description:	Generic search alias for products
Type of rule:	Substitution
Incoming URL pattern:	/find/*
Replacement pattern:	/allsearch.nsf/products?searchview&query=*

Armed with this rule, you can create simple HTML links like this:

What's on Sale

which the server will transform to:

/allsearch.nsf/products?searchview&query=deals.

Wildcards in substitution rules
The incoming and replacement patterns in substitution rules *must* each specify at least one wildcard. If you do not explicitly include a wildcard somewhere in a pattern, the HTTP task will automatically append "/*" to the pattern when it puts the rule in its internal table. For example, the following rule:

Incoming URL pattern:	/shopping
Replacement pattern:	/market

is internally equivalent to:

Incoming URL pattern:	/shopping/*

Replacement pattern:	/market/*

A pattern that ends in "/*" (whether the ending wildcard was entered explicitly or added by Domino) will always match an incoming URL that lacks an ending slash but otherwise matches the pattern. This insures that a rule that is intended to remap a directory will match URLs that end with that directory. For example, the rule above will map incoming URLs as follows:

- /shopping/petstore maps to /market/petstore
- /shopping/ maps to /market/
- /shopping maps to /market

Redirection rules

As mentioned previously, there are two types of redirection rules: external redirection and internal redirection. External redirection rules are used when the location of a Web site resource has changed and you want the browser to be aware of the new location. Internal redirection rules are used when you do *not* want the browser to be aware of the new location (but please note that the purpose of doing so is for site navigation, not security; as I stated emphatically above, never rely on any type of rule for site security).

The two types of redirection rules are distinguished by the Redirect to this URL field in the Web Site Rule document. If the pattern in this field starts with a slash (for example, /welcome.nsf), the rule is treated as an internal redirection. Otherwise, the rule is assumed to be an external redirection. The pattern for an external redirection needs to start with an Internet protocol string that the browser will understand—usually http: or https:, but it can be another protocol such as ftp:.

Wildcards are allowed in redirection rules, but are not required. Remember that at least one wildcard is required for substitution rules and that Domino 6 will add "/*" to any substitution rule pattern that doesn't have an explicit wildcard. Domino 6 doesn't do this for redirection rules because there are cases when you want to force an exact match on the URL path. We will discuss a common example of this below.

External redirection

An external redirection rule causes the server to inform the browser that a file or other resource requested by the browser is now located at some other URL. The browser will then automatically issue another request with the new URL. This is largely transparent to the browser user, except that the URL in the address box of the browser will change to reflect the new URL. Also, if the user bookmarks the page, the bookmark will use the new URL.

Example 1: External redirection when moving a Web site

External redirection rules are often used when all or part of a Web site has moved to another (presumably related) site. Using external redirections allows existing links and bookmarks to keep working, but insures that new bookmarks will point to the new location.

For example, suppose you are the Web administrator for SpendBucks.com, which specializes in hosting virtual stores. One of your clients, Fluffy Pets, Inc., has decided that it wants you to significantly upgrade their store, currently located at http://SpendBucks.com/FluffyPets. As part of this upgrade you decide to give Fluffy Pets their own domain name "fluffypets.com," which you assign to your hosting server. However, many of Fluffy Pets' existing customers may have bookmarked the old site. In addition, it will take some time to change all of Fluffy Pets' current banner ads and other advertising media to point to the new site. How do you ensure that people following an outdated link will find the new site? To provide a smooth transition, you can create an external redirection rule:

Description:	FluffyPets new site
Type of rule:	Redirection
Incoming URL pattern:	/fluffypets/*
Redirect to this URL:	http://fluffypets.com/*

Now, if a customer clicks a bookmark or banner ad that is set to

http://spendbucks.com/fluffypets/welcome.nsf

the server will redirect the browser to:

http://fluffypets.com/welcome.nsf

If the original URL includes a query string, it will automatically be appended to the redirected URL. For example, if the incoming URL is http://spendbucks.com/fluffypets/products.nsf?Open, then the redirected URL will be http://fluffypets.com/products.nsf?Open. You can also override the original query string by including a new query string in the Redirect to this URL field pattern.

Example 2: External redirection to change protocols
As mentioned above, an external redirection can start with any protocol string. Thus, you can use an external redirection to change the protocol of a request. A common scenario is to redirect the user from a nonsecure site (http:) to an SSL site (https:). For example:

Description:	Redirect to secure site for registration
Type of rule:	Redirection
Incoming URL pattern:	/*/register.htm
Redirect to this URL:	https://secure.spendbucks.com/*/register.htm

With this rule, when a customer wants to register at any of the virtual stores on spendbucks.com, they will be redirected to the SSL site. For example, if the user clicks on the hotspot:

<a href="/gasguzzler/register.htm">Tell Us Everything About Youself!<\a>

the server will redirect the browser to the secure site:

https://secure.spendbucks.com/gasguzzler/register.htm

Example 3: External redirection to change ports
You can also use an external redirection to redirect to another port on the target site. (Often a target site is the same as the original site, just a different port). For example:

Description:	Account application is at port 8008
Type of rule:	Redirection
Incoming URL pattern:	/account.nsf/*
Redirect to this URL:	http://spendbucks.com:8008/account.nsf/*

A few technical details
In technical terms, an external redirection causes the server to return an HTTP status code of 302 and a Location header with the new URL. All current browsers support external redirections, but some very old browsers may not recognize the 302 status and may display a blank page or an error to the user.

There is one very important implication of external redirection that you need to keep in mind: If you redirect a request that includes data (such as the user filling out a form), it is up to the browser to decide whether or not to forward the data to the redirected site. Because of this, it is safest to use external redirections only for HTTP requests that use the GET or HEAD methods. If you want to use a redirection for other methods such as POST (for submitting a form) or the WebDAV methods, you should carefully test the redirections with your clients' browsers.

Also, Domino 6 leaves the interpretation of the new URL entirely up to the browser; it doesn't try to validate its syntax. All that Domino 6 looks for is whether or not the redirection pattern starts with a slash. If it doesn't, the redirection will be sent to the browser. Thus, if you use anything other than the standard protocols, it is once again up to you to test the redirection with your clients' browsers.

Internal redirection
An internal redirection rule acts very much like a substitution rule in that the HTTP task generates a new URL and re-normalizes it. However, there are two differences. First, the redirection table is searched recursively, so you can "nest" redirection rules. Second, an internal redirection does not require a wildcard, so you can use it instead of a substitution rule if you want to force an exact match on the URL path. By the way, when we talk about an "exact match," remember that rule-matching is done on the fully normalized path and that it is not case-sensitive.

There aren't too many scenarios where you need to force an exact match. However, one scenario is so common it is actually specified as a option on the Web Site document itself. If a browser user just wants to go to the home page of a Web site, the user typically enters the host name into the address bar of the browser, for example www.dominoinc.com. The browser will actually convert this to the minimal URL of http://www.dominoinc.com/ (you may actually see the browser update the address bar with the URL). In the converted URL, the path is just "/". So, the target server needs to interpret a URL path of "/" as meaning "go to the home page."

In Domino 6, you specify the home page with the Home URL field on the Configuration tab of the Web Site document. The Domino 6 HTTP task automatically builds a redirection rule using this field. For example, if you set this field to /welcome.nsf, then Domino 6 constructs a redirection rule equivalent to this:

Description:	
Type of rule:	Redirection
Incoming URL pattern:	/
Redirect to this URL:	/welcome.nsf

Notice that Domino 6 *must* build a redirection rule, not a substitution rule, because substitution rules always assume a wildcard and you only want the rule to be invoked if the URL path is exactly "/". You don't want the rule invoked for anything else like /cgi-bin/contact.pl or /products.nsf.

Although wildcards aren't required in internal redirections, you are allowed to use them in both the Incoming URL pattern and Redirect to this URL fields. However, there isn't much point in doing so; you might as well just use a substitution rule. Also, internal redirections can't be used for any request methods other than GET or HEAD.

Migrating from Domino R5

If you have an existing R5 Web server, you may already have rules defined in the Domino Directory Server\Web Configurations view. If you plan to upgrade that server to Domino 6, you may wonder how the existing rules will be interpreted. Domino 6 supports the same set of rule types as R5, but the terminology has been changed to be less confusing. Here are the old and new terms:

R5	Domino 6
URL->URL	Substitution
URL->Redirection	Redirection
URL->Directory	Directory

The Domino 6 HTTP task handles rules in a much more systematic way than R5. In R5, all three types of rules are placed into a single table in no particular order (roughly the order in which they were created, but that isn't strictly reliable). In Domino 6, each rule type has its own table and is ordered in decreasing length of the Incoming URL pattern. This makes the application of rules in Domino 6 much more obvious.

The Domino 6 HTTP task can use either of the two Domino Directory views: the old Server\Web Configurations view or the new Server\Internet Sites view. If your Domino 6 server uses the old view, it still places the rules in separate tables, in order. Normally this won't be a concern, but if you have a lot of rules or have rules with very similar patterns, you should carefully test all the rules after upgrading.

In this article we've covered two of the three types of rules, substitution and redirection, and given examples about how they can make your site easier to organize. Stay tuned to LDD for a future article about directory rules.

About the author
John Chamberlain is a Senior Software Engineer on the Domino 6 Web Server team.

