Working with URLs

A URL is a container for a file (or other resource), which may either be on the same system the application is running on, or on another system that's accessible via the Internet.

This topic discusses the various URL schemes that LiveCode implements, how to create and manipulate files using URLs, and how to transfer data between your system and an FTP or HTTP server.

To fully understand this topic, you should know how to create objects and write short scripts, and understand how to use variables to hold data. You should also have a basic understanding of how the Internet works.

An Overview of URLs

In the LiveCode language, a URL is a container for a `file` or other document, such as the output of a CGI on a web server. The data in a URL may be on the same system the application is running on, or may be on another system.

URLs in LiveCode are written like the URLs you see in a browser. You use the **URL** keyword to designate a URL, enclosing the URL's name in double quotes:

```
put field "Info" into URL "file:myfile.txt"
get URL "http://www.example.org/stuff/nonsense.html"
put URL "ftp://ftp.example.net/myfile" into field “Data"
```

URL Schemes

A URL scheme is a type of URL. LiveCode supports five URL schemes with the **URL** keyword: **http**, **ftp**, **file**, **binfile**, and (for backwards compatibility on Mac OS X) **resfile**.

The **http** and **ftp** schemes designate documents or directories that are located on another system that's accessible via the Internet. The **file**, **binfile**, and **resfile** schemes designate local files.

The http scheme

An **http** URL designates a document from a web server:

```
put URL "http://www.example.org/home.htm" into field "Page"
```

When you use an `http` URL in an expression, LiveCode downloads the URL from the server and substitutes the downloaded data for the URL.

When you put something into an `http` URL, LiveCode uploads the data to the web server:

```
put field "Info" into URL “http://www.example.net/info.htm"
```

Note: Because most web servers do not allow `http` uploads, putting something into an `http` URL usually will not be successful. Check with the server's administrator to find out whether you can use the `http` protocol to upload files.

For more details about `http` URLs, see the entry for the `http` keyword in the LiveCode Dictionary.

The ftp scheme

An **ftp** URL designates a file or directory on an FTP server:

```
get URL "ftp://user:passwd@ftp.example.net/picture.jpg"
```

When you use an `ftp` URL in an expression, LiveCode downloads the URL from the server and substitutes the downloaded data for the URL. When you put something into an ftp URL, LiveCode uploads the data to the ftp server:

```
put image 10 into URL
    “ftp://user:passwd@ftp.example.net/picture.jpg"
```

FTP servers require a user name and password, which you can specify in the URL. If you don't specify a user name and password, LiveCode adds the "anonymous" user name and a dummy password automatically, in accordance with the convention for public FTP servers.

Note: Uploading to an FTP server usually requires a registered user name and password.

For more details about `ftp` URLs, see the entry for the `ftp` keyword in the LiveCode Dictionary.

Directories on an FTP server

A URL that ends with a slash (/) designates a directory (rather than a file). An `ftp` URL to a directory evaluates to a listing of the directory's contents.

The file scheme

A **file** URL designates a file on your system:

```
put field "Stuff" into URL "file:/Disk/Folder/testfile"
```

When you use a `file` URL in an expression, LiveCode gets the contents of the `file` you designate and substitutes it for the URL. The following example puts the contents of a `file` into a variable:

```
put URL "file:myfile.txt" into myVariable
```

When you put data into a `file` URL, LiveCode puts the data into the file:

```
put myVariable into URL “file:/Volumes/Backup/data"
```

Note: As with local variables, if the file doesn't exist, putting data into it creates the file.

To create a URL from a file path that LiveCode provides, use the **&** operator:

```
answer file "Please choose a file to get:"
get URL ("file:" & it)
```

File path syntax and the file scheme:

The `file` URL scheme uses the same `file` path syntax used elsewhere in LiveCode statements. You can use both absolute paths and relative paths in a `file` URL.

Conversion of end-of-line markers

Different operating systems use different characters to mark the end of a line. Mac OS X uses a return character (ASCII 13), Linux systems use a linefeed character (ASCII 10), and Windows systems use a return followed by a linefeed. To avoid problems when transporting a stack between platforms, LiveCode always uses linefeeds internally when you use a `file` URL as a container. LiveCode translates as needed between the your system's end-of-line marker and LiveCode's linefeed character. To avoid this translation, use the `binfile` scheme (see below).

The binfile scheme

A **binfile** URL designates a file on your system that contains binary data:

```
put URL "binfile:beachball.gif" into image “Beachball"
```

When you use a **binfile** URL in an expression, LiveCode gets the contents of the file you designate and substitutes it for the URL.

The following example puts the contents of a file into a variable:

```
put URL "binfile:picture.png" into pictVar
```

When you put data into a **binfile** URL, LiveCode puts the data into the file:

```
put pictVar into URL "binfile:/Volumes/Backup/pict.png"
put image 1 into “binfile:/image.png"
```

As with local variables, if the file doesn't exist, putting data into it creates the file.

The **binfile** scheme works like the file scheme, except that LiveCode does not attempt to convert end-of-line markers. This is because return and linefeed characters can be present in a binary file but not be intended to mark the end of the line. Changing these characters can corrupt a binary file, so the **binfile** scheme leaves them alone.

The resfile scheme
On Mac OS Classic (and sometimes on OS X systems), files can consist of either a data fork or a resource fork or both.

Important: While LiveCode supports reading and writing resource fork files on Mac OS X, this feature is only intended to help you access and work with legacy files. We do not generally recommend the use of resource forks when designing any new application.

The resource fork contains defined resources such as icons, menu definitions, dialog boxes, fonts, and so forth. A **resfile** URL designates the resource fork of a Mac OS X file:

```
put myBinaryData into URL “resfile:/Disk/Resources"
```

When you use a **resfile** URL in an expression, LiveCode gets the resource fork of the file you designate and substitutes it for the URL.

When you put data into a **resfile** URL, LiveCode puts the data into the file's resource fork.

Note: A **resfile** URL specifies the entire resource fork, not just one resource. To work with individual resources, use the **getResource**, **setResource**, **deleteResource** and **copyResource** functions.

The most common use for this URL scheme is to copy an entire resource fork from one file to another. To modify the data from a **resfile** URL, you need to understand the details of Apple's resource fork format.

Creating a resource fork
Unlike the **file** and **binfile** URL schemes, the **resfile** keyword cannot be used to create a file. If the file doesn't yet exist, you cannot use the **resfile** keyword to create it. To create a new

resource file, first use a **file** URL to create the file with an empty data fork, then write the needed data to its resource fork:

```
put empty into URL "file:myFile" -- creates an empty file
put myStoredResources into URL “resfile:myFile"
```

Manipulating URL contents

You use a URL like any other container. You can get the content of a URL or use its content in any expression. You can also put any data into a URL.

http, **ftp**, **binfile**, and **resfile** URLs can hold binary data.

http, **ftp**, and **file** URLs can hold text.

The URL keyword
To specify a URL container, you use the **URL** keyword before the URL, which can use any of the five schemes described above:

```
if URL "http://www.example.net/index.html" is not empty then ...
```

```
get URL "binfile:/Applications/Hover.app/data"
```

```
put 1+1 into URL “file:output.txt"
```

The `URL` keyword tells LiveCode that you are using the `URL` as a container.

Some properties (such as the **filename** of a player or image) let you specify a URL as the property's value. Be careful not to include the **URL** keyword when specifying such properties, because using the URLkeyword indicates that you're treating the

URL as a container. If you use the URL keyword when specifying such a property, the property is set to the contents of the URL, not the URL itself, and this is usually not what's wanted.

Using the content of a URL

As with other containers, you use the content of a URL by using a reference to the URL in an expression. LiveCode substitutes the URL's content for the reference.

If the URL scheme refers to a local file (**file**, **binfile**, or **resfile** URLs), LiveCode reads the content of the file and substitutes it for the URL reference in the expression:

```
answer URL "file:../My File"
-- displays the file's content
put URL "binfile:flowers.jpg" into myVariable
put URL "resfile:Icons" into URL "resfile:New Icons”
```

If the URL scheme refers to a document on another system (**http** or **ftp** URLs), LiveCode downloads the URL automatically, substituting the downloaded data for the URL reference:

```
answer URL “http://www.example.net/files/greeting.txt"
```

Note: If the server sends back an error message--for example, if the file you specify in an **http** URL doesn't exist--then the error message replaces the URL reference in the expression.

Important: When you use an **ftp** or **http** URL in an expression, the handler pauses until LiveCode is finished downloading the URL. If you do not want to block LiveCode when accessing these resources, use the **load URL** form of the command (see below).

Putting data into a URL

As with other containers, you can put data into a URL. The result of doing so depends on whether the URL scheme specifies a file on your system (**file**, **binfile**, or **resfile**) or on another system (**http** or **ftp**).

If the URL scheme refers to a local file (**file**, **binfile**, or **resfile** URLs), LiveCode puts the data into the specified file:

```
put field "My Text" into URL "file:storedtext.txt"
put image 1 into URL “binfile:picture.png"
```

If the URL scheme refers to a document on the Internet (**http** or **ftp** URLs), LiveCode uploads the data to the URL:

```
put myVar into URL “ftp://me:pass@ftp.example.net/file.dat"
```

Because most web servers do not allow HTTP uploads, this usually will not be successful with the **http** scheme.

Chunk expressions and URLs

Like other containers, URLs can be used with chunk expressions to specify a portion of what's in a URL--a line, an item, a word, or a character. In this way, any chunk of a URL is like a container itself.

For more information about Chunk Expressions, see the guide on *Processing Text and Data*.

You can use any chunk of a URL in an expression, in the same way you use a whole URL:

```
get line 2 of URL "http://www.example.net/index.html"
put word 8 of URL "file:/Disk/Folder/myfile" into field 4
if char 1 of URL "ftp://ftp.example.org/test.jpg" is "0" then …
```

You can also specify ranges, and even one chunk inside another:

```
put char 1 to 30 of URL "binfile:/marks.dat" into myVar
answer line 1 to 3 of URL “http://www.example.com/file"
```

Putting data into a chunk

If the URL is local (that is, if it is a **file**, **binfile**, or **resfile** URL), you can put a value into a chunk of the URL:

```
put it into char 7 of URL "binfile:/picture.gif" put return after
    word 25 of URL "file:../datafile"
put field 3 into line 20 of URL "file:myfile.txt"
```

You can also put a value into a chunk of an **ftp** or **http** URL.

Because it's impossible to upload part of a file, LiveCode downloads the file, makes the change, then uploads the file back to the server.

Tip: This method is inefficient if you need to make several changes. In this case, it's faster to first put the URL in a variable, replace the chunk you want to change, then put the variable into the URL:

```
put URL "ftp://me:secret@ftp.example.net/file.txt" into myVar
put field "New Info" after line 7 of myVar
put field "More" into word 22 of line 3 of myVar
put myVar into URL "ftp://me:secret@ftp.example.net/file.txt"
```

This ensures that the file only needs to be downloaded once and re-uploaded once, no matter how many changes you need to make.

****URLs and memory****
URLs, unlike other containers, are only read into memory when you use the URL in a statement. Other containers – like variables, fields, buttons, and images – are normally kept in memory, so accessing them doesn't increase memory usage.

This means that in order to read a URL or place a value in a chunk of a URL, LiveCode reads the entire file into memory.

Because of this, you should be cautious when using a URL to refer to any very large file.

Even when referring to a single chunk in a URL, LiveCode must place the entire URL in memory.

An expression such as line 347882 of URL "file:bigfile.txt" may be evaluated very slowly or even not work at all, if insufficient memory is available.

If you refer to a chunk of an `ftp` or http URL, LiveCode must download the entire file to find the chunk you specify.

If you need to read and write large quantities of data to a file, or seek through the contents of a large file without loading the entire contents into memory, use the **open file**, **read from file**, **seek** and **close file** commands instead of the URL commands.

For more information on these commands see the *LiveCode Dictionary*.

Deleting URLs

You remove a URL with the **delete URL** command.

To delete a local file, you use a **file** or **binfile** URL:

```
delete URL "file:C:/My Programs/test.exe"
delete URL"binfile:../mytext.txt"
```

It doesn't matter whether the file contains binary data or text; for deletion, these URL schemes are equivalent.

Tip: You can also use the **delete file** command to remove a file. To delete the resource fork of a file, you use a **resfile** URL. The following example removes the resource fork along with all resources, but leaves the file in place:

```
delete URL "resfile:/Volumes/Backup/proj.rev"
```

Tip: To delete a single resource instead of the entire resource fork, use the **deleteResource** function.

To remove a file or directory from an FTP server, you use an **ftp** URL:

```
delete URL "ftp://root:secret@ftp.example.org/deleteme.txt"
delete URL "ftp://me:mine@ftp.example.net/trash/"
```

As with creating files, you can use an **http** URL to delete a file, but most HTTP servers are not configured to allow this.