Byte ordering refers to the order in which a multibyte datum is written to a file. For example, a 16-bit word (sometimes called a "short") consists of a high-order byte and a low-order byte. Under big-endian byte ordering, which is commonly used on Macintosh, the high-order byte is written to the file first. Under little-endian byte ordering, which is commonly used on Windows, the low-order byte is written to the file first.

FBinWrite will write an entire structure to a disk file. The individual fields of the structure will be byte-swapped if the /B flag is designated.

The FBinWrite operation is not multidimensional aware. See **Analysis on Multidimensional Waves** on page II-95 for details.

See Also

FBinRead, **Open**, **FGetPos**, **FSetPos**, **FStatus**, **GBLoadWave**

FetchURL

FetchURL(*urlStr*)

The FetchURL function returns a string containing the server's response to a request to get the contents of the URL specified by *urlStr*. If *urlStr* contains a URL that uses the file:// scheme, the contents of the local file is returned.

Parameters

urlStr is a string containing the URL to retrieve. You can include a username, password, and server port number as part of the URL.

FetchURL expects that *urlStr* has been percent-encoded if it contains reserved characters. See **Percent Encoding** on page IV-268 for additional information on when percent-encoding is necessary and how to do it.

See **URLs** on page IV-267 for details about how to correctly specify the URL.

FetchURL supports only the http://, https://, ftp://, and file:// schemes. See **Supported Network Schemes** on page IV-268 for details.

There are two special values of *urlStr* that can be used to get information about the network library that Igor uses. The keyword=value pairs returned when *urlStr* is "curl_version_info" may be useful to programmers in that the features and protocols available in the library are specified.

```
FetchURL("curl_version")
FetchURL("curl_version_info")
```

Details

If FetchURL encounters an error, it returns a NULL string. You should check for errors before using the returned string. In a user-defined function, use the **GetRTError** function.

```
String urlStr = "http://www.badserver"
String response = FetchURL(urlStr)
Variable error = GetRTError(1)       // Check for error before using response
if (error != 0)
    // FetchURL produced an error
    // so don't try to use the response.
endif
```

Limitations

It is possible for FetchURL to return a valid server response even though the URL you requested does not exist on the server or requires a username and password that you did not provide. In this situation, the response returned by the server will usually be a web page stating that the page was not found or another error message. You can check for this kind of error in your own code by examining the response.

FetchURL does not support advanced features such as network proxies, file or data uploads, setting the timeout period, or saving the server's response directly to a file. When using the http:// scheme, only the GET method is supported. This means that you cannot use FetchURL to submit form data to a web server that requires using the http POST method. Use the **URLRequest** operation if you need any of these features.

Igor Pro is not capable of displaying the contents of a URL in a rendered form like a web browser.

Examples

```
// Retrieve the contents of the WaveMetrics home page.
String response
```