```
   String url = ""
   sprintf url, "%s?query=%s", baseURL, keywords

   // Fetch the results.
   String response
   response = FetchURL(url)
   Variable error = GetRTError(1)
   if (error != 0 || numtype(strlen(response)) != 0)
      Print "Error fetching search results."
      return -1
   endif

   // Try to extract the thumbnail image of the first result.
   String regExp = "(?s)<img class=\"cover-thumb\" src=\"(.+?)\".*"
   String firstURL
   SplitString/E=regExp response, firstURL
   firstURL = TrimString(firstURL)
   firstURL = "http:" + firstURL
   if (V_flag == 1)
      BrowseURL firstURL
   else
      Print "Could not extract the first result from the"
      Print "results page. Your search terms might not"
      Print "have given any results, or the format of"
      Print "the results may have changed so that the"
      Print "first result cannot be extracted."
      return -1
   endif

   return 0
End
```

Examples using the POST method can be found in the section ***The HTTP POST Method*** of the documentation for the **URLRequest** operation.

HTTP Troubleshooting

Here are some tips if you experience errors using **FetchURL** or **URLRequest**:

1. Use a web browser to connect to the site. This confirms that your network is operating, the server is operating, and that you are using the correct URL.
2. FetchURL and URLRequest generate an error if it cannot connect to the destination server, which could happen if your computer is not connected to the network or if the target URL contains an invalid host name or port number.

   However if the URL contains an invalid path or if the destination URL requires you to provide a username and password, these operations will likely not generate an error. The reason is these errors typically result in a web page being returned, though not the one you expected. If you need to check that a call to FetchURL returned a valid web page and not an error web page, you must do that in your own code. One possibility would be to try searching the page for key phrases, such as "File Not Found" or "Page Not Found".

   With URLRequest, you can inspect the value of the V_responseCode output variable. For successful HTTP requests, this value will usually be 200. A different value may indicate that there was an error making the request.

Operation Queue

Igor implements an operation queue that supports deferred execution of commands and some special commands for dealing with files, procedures, and experiments.

Igor services the operation queue when no procedures are running and the command line is empty. If the operation queue is not empty, Igor then executes the oldest command in the queue.

You can append a command to the operation queue using

```
Execute/P <command string>
```

The /P flag tells Igor to post the command to operation queue instead of executing it immediately.

You can also specify the /Q (quiet) or /Z (ignore error) flags. See **Execute/P** operation (page V-204) for details about /Q and /Z.

The command string can contain either a special command that is unique to the operation queue or ordinary Igor commands. The special commands are:

INSERTINCLUDE *procedureSpec*

DELETEINCLUDE *procedureSpec*

AUTOCOMPILE ON

AUTOCOMPILE OFF

COMPILEPROCEDURES

NEWEXPERIMENT

LOADFILE *filePath*

LOADFILEINNEWINSTANCE *filePath*

LOADHELPEXAMPLE *filePath*

MERGEEXPERIMENT *filePath*

RELOAD CHANGED PROCS (added in Igor Pro 9.00)

Note: The special operation queue keywords must be all caps and must have exactly one space after the keyword.

INSERTINCLUDE and DELETEINCLUDE insert or delete #include lines in the main procedure window. *procedureSpec* is whatever you would use in a #include statement except for "#include" itself.

AUTOCOMPILE ON and AUTOCOMPILE OFF require Igor Pro 8.03 or later. See **Auto-Compiling** on page III-404 for details.

COMPILEPROCEDURES does just what it says, compiles procedures. You must call it after operations such as INSERTINCLUDE that modify, add, or remove procedure files.

NEWEXPERIMENT closes the current experiment without saving.

LOADFILE opens the file specified by *filePath*. *filePath* is either a full path or a path relative to the Igor Pro Folder. The file can be any file that Igor can open. If the file is an experiment file, execute NEWEXPERIMENT first to avoid displaying a "Do you want to save" dialog. If you want to save the changes in an experiment before loading another, you can use the standard SaveExperiment operation.

LOADFILEINNEWINSTANCE opens the file specified by *filePath* in a new instance of Igor. *filePath* is either a full path or a path relative to the Igor Pro Folder. The file can be any file that Igor can open. LOADFILEINNEWINSTANCE was added in Igor Pro 7.01.

LOADHELPEXAMPLE opens the file specified by *filePath* in a new instance of Igor. It works the same as LOADFILEINNEWINSTANCE and is recommended for **Notebook Action Special Characters** that load example experiments, such as those used throughout Igor's help files. LOADHELPEXAMPLE was added in Igor Pro 8.00.

MERGEEXPERIMENT merges the experiment file specified by *filePath* into the current experiment. Before using this, make sure you understand the caveats regarding merging experiments. See **Merging Experiments** on page II-19 for details.