

## Simple Use Case

```
var callback = {
  success: function(o) {
    document.getElementById('someEl').innerHTML =
      o.responseText;
  }
}

var connectionObject =
  YAHOO.util.Connect.asyncRequest('GET', 'file.php',
    callback);
```

Executes an asynchronous connection to `file.php`. If the HTTP status of the response indicates success, the full text of the HTTP response is placed in a page element whose ID attribute is "someEl".

## Invocation (asyncRequest)

```
YAHOO.util.Connect.asyncRequest(str http method, str url[,
  obj callback object, str POST body]);
```

**Arguments:**

- (1) **HTTP method (string):** GET, POST, HEAD, PUT, DELETE, etc. PUT and DELETE are not supported across all A-Grade browsers.
- (2) **URL (string):** A url referencing a file that shares the same server DNS name as the current page URL.
- (3) **Callback (object):** An object containing success and failure handlers and arguments and a scope control; see Callback Object detail for more.
- (4) **POST body (string):** If you are POSTing data to the server, this string holds the POST message body.

**Returns: Transaction object.** { `tId`: int *transaction id* } The transaction object allows you to interact (via Connection Manager) with your XHR instance; pass `tId` to CM methods such as `abort()`.

## Callback Object: Members (All Optional)

1. **customevents:** Object containing any Custom Event handlers for transaction-level events (as alternatives to *success* and *failure* handlers below). Transaction-level Custom Events include *onStart*, *onComplete*, *onSuccess*, *onFailure*, *onAbort* and receive the same arguments as their global counterparts (see Global Custom Events, above right).
2. **success (fn):** The success method is called when an `asyncRequest` is replied to by the server with an HTTP in the 2xx range; use this function to process the response.
3. **failure (fn):** The failure method is called when `asyncRequest` gets an HTTP status of 400 or greater. Use this function to handle unexpected application/communications failures.
4. **argument (various):** The argument member can be an object, array, integer or string; it contains information to which your success and failure handlers need access.
5. **scope (obj):** The object in whose scope your handlers should run.
6. **timeout (int):** Number of milliseconds CM should wait on a request before aborting and calling failure handler.
7. **upload (fn):** Handler to process file upload response.

## Global Custom Events

These events fire for all transactions; subscribe via `YAHOO.util.Connect`; e.g.: `YAHOO.util.Connect.startEvent.subscribe(myFn);`

Event	Fires when...	Arguments
startEvent	transaction begins	transaction ID
completeEvent	transaction complete, but not yet reconciled as success or failure	transaction ID
successEvent	HTTP 2xx response received	Response object
failureEvent	HTTP 4xx, 5xx, or unknown response received	Response object
abortEvent	timeout/abort succeeds	transaction ID

## Response Object

Your **success**, **failure**, and **upload** handlers are passed a single argument; that argument is an object with the following members:

tId	The transaction id.
status	The HTTP status code of the request.
statusText	The message associated with the HTTP status.
getResponseHeader[]	Array collection of response headers and their corresponding values, indexed by header label.
getAllResponseHeaders	String containing all available HTTP headers with name/value pairs delimited by "\n".
responseText	The server's full response as a string; for upload, the contents of the response's <code>&lt;body&gt;</code> tag.
responseXML	If a valid XML document was returned and parsed successfully by the XHR object, this will be the resulting DOM object.
argument	The arguments you defined in the Callback object's <code>argument</code> member.

## Solutions

**Roll up an existing form on the page**, posting its data to the server:

```
YAHOO.util.Connect.setForm('formId');
var cObj = YAHOO.util.Connect.asyncRequest('POST',
  'formProcessor.php', callback);
```

**Cancel a transaction in progress:**

```
//if the transaction is created as follows...
var cObj = YAHOO.util.Connect.asyncRequest('GET',
  myServer.php', callback);
//...then you would attempt to abort it this way:
YAHOO.util.Connect.abort(cObj);
```

Connection Manager sets headers automatically for GET and POST transactions. If you need to **set a header manually**, use this syntax:

```
YAHOO.util.Connect.initHeader('SOAPAction', 'myAction');
```

## Dependencies

Connection Manager requires the YAHOO Global Object and the Event Utility.

## Key methods of YAHOO.util.Connect:

(o = Transaction object)

**abort(o)**

**asyncRequest()**  
**initHeader(s label,**  
**s value, [b**

**persistHeader])** optional  
param persists header as a  
default for each subsequent  
transaction.

**isCallInProgress(o)**

**setForm(str formId | o**  
**form el ref, b**  
**isUpload, s**

**secureUri])** optional params  
for file upload only; provide  
secureUri for iFrame only under  
SSL

**setPollingInterval(int i)**

**setProgId(id)**

## HTTP Status Codes

2xx	Successful
3xx	Redirection
4xx	Client error
5xx	Server error

0	Communication failure
200	OK
400	Bad request
401	Unauthorized
403	Forbidden
404	Not found
408	Request timeout
410	Gone
500	Internal server error
502	Bad gateway
503	Service unavailable