

# HTTP\_Services

Application service module that provides several core services to help with HTTP request and HTTP response management.

## Syntax

```
Response = HTTP_Services(@Service, @Params)
```

## Returns

The meaning of the response value depends on the service.

## Parameters

Parameter	Description
@Service	The name of the service being requested. <b>Required.</b>
@Params	Generic parameters. Refer to a specific service to determine the actual parameters used.

## Remarks

This service module contains the primarily library of critical HTTP communication services. In here are the services that process the HTTP request, sets the relevant HTTP request header fields, sets the HTTP method, and then allows the developer to set the appropriate HTTP response header fields, status code, and body. Many of these services are called automatically from core routines (like the [HTTP\\_MCP controller](#)) so that the critical HTTP request details are accessible to the developer in a very convenient manner.

## Services

Service	Description
<b>RunWebAPI</b>	Calls the web API procedure for the resource associated to the URL endpoint being requested. Note, when this service is called normally the arguments are empty. They only exist for testing a web API without having to run from an actual HTTP request.
<a href="#">RunHTTPS service</a>	Calls the indicated HTTP web service routine.
<a href="#">SetSelfURL</a>	Sets the self URL for the current service. The self URL is the URL that identifies itself. It is typically returned in responses to serve as a self-referencing ID apart from other URLs that might be returned which direct the caller to other services.
<a href="#">GetSelfURL</a>	Returns the self URL for the current service.
<a href="#">SetSessionID</a>	Creates and sets a unique Session ID for the current HTTP Request/Response. This is used to stamp various logs.
<a href="#">GetSessionID</a>	Returns the unique Session ID for the current HTTP Request/Response. This is used to stamp various logs.
<a href="#">CreateLogFile</a>	Creates a log file in the designated capture path.
<a href="#">SetOE CGI Request</a>	Sets the HTTP request that the OE CGI creates to memory so it can be retrieved by later routines. This avoids the need to pass this around into various routines.
<a href="#">GetOE CGI Request</a>	Returns the original HTTP request that the OE CGI creates.
<a href="#">SetOE CGI ProcErr</a>	Sets the HTTP ProcErr that the OE CGI creates to memory so it can be retrieved by later routines. This avoids the need to pass this around into various routines.
<a href="#">GetOE CGI ProcErr</a>	Returns the original HTTP ProcErr that the OE CGI creates.
<a href="#">SetHTTPV alue</a>	Sets a specific HTTP request value. This is normally set within the <a href="#">SetOE CGI Request</a> service and done directly using <a href="#">Memory_Services</a> for efficiency, but some APIs might need to override an HTTP request value.

<b>GetHTTPV alue</b>	Returns a specific HTTP request value. This is normally set within the <a href="#">SetOECGIRequest</a> service. This method is not called directly as other services normally are. It is a generic service that various 'GetHTTPxxxx' services will call. This is why the meta data doesn't include GetHTTPValue but one or more specific GetHTTPxxxx options.
<b>SetRequestHeaderFields</b>	Sets all of the Request Header Fields based on the content of the HTTP request that the OECGI creates. This assumes the <a href="#">SetOECGIRequest</a> service has already been called so that the Request array is in memory.
<b>SetRequestHeaderField</b>	Sets the indicated Request Header Field with the indicated value. This can then be retrieved with a <a href="#">GetRequestHeaderField</a> service call so that server processing can operate accordingly.
<b>GetRequestHeaderFields</b>	Returns all of the Request Header Field names and values. These are formatted as Name : <space> Value <crLf> with an extra <crLf> appended after the last field/value pair.
<b>GetRequestHeaderField</b>	Returns the value previously set for the indicated Request Header Field. The Name argument is case-insensitive but if the indicated Request Header Field has not been set then an error condition will be set.
<b>SetQueryFields</b>	Sets all of the Query Fields based on the content of the HTTP request that the OECGI creates. This assumes the <a href="#">SetOECGIRequest</a> service has already been called so that the Request array is in memory.
<b>SetQueryField</b>	Sets the indicated Query field with the indicated value. This can then be retrieved with a <a href="#">GetQueryField</a> service call so that server processing can operate accordingly.
<b>GetQueryField</b>	Returns the value previously set for the indicated Query Field. If then indicated Query Field has not been set then then an error condition will be set.
<b>SetResponseHeaderField</b>	Sets the indicated Response Header Field with the indicated value. This can then be retrieved with a <a href="#">GetRequestHeaderField</a> service call so that server processing can operate accordingly.
<b>SetCookie</b>	Adds a Set-Cookie header to the response using the indicated Name. The cookie's value and optional attributes will automatically be included as indicated by each argument.
<b>GetResponseHeaderFields</b>	Returns all of the Response Header Field names and values. These are formatted as Name : <space> Value <crLf> with an extra <crLf> appended after the last field/value pair. This also returns the response status since the CGI specification uses the "Status" header field. This will be put into the response before the regular header field/values.
<b>GetResponseHeaderField</b>	Returns the value previously set for the indicated Response Header Field. The Name argument is case-insensitive but if the indicated Response Header Field has not been set then it an error condition will be set.
<b>SetResponseStatus</b>	Sets the HTTP status code to be used in the response. This can be set at any time. If unchanged by other routines this will be the status code used.
<b>GetResponseStatus</b>	Gets the current HTTP status code to be used in the response. If this has not yet been set then code 200 (OK) will be used as a default.
<b>SetResponseBody</b>	Sets the body content to be returned in the response.
<b>GetResponseBody</b>	Gets the current body to be returned in the response.
<b>GetResponseBodyIsBinary</b>	Gets the binary flag associated with the current body. This is normally set by the <a href="#">SetResponseBody</a> service at the time the body content is also set.
<b>GetResponse</b>	Gets the full response, headers and body, to be returned to the HTTP request. This will build the response as needed by OECGI, especially regarding binary data handling.
<b>SetResponseError</b>	Sets a response error. This follows the RFC 7807 specification ( <a href="https://tools.ietf.org/html/rfc7807">https://tools.ietf.org/html/rfc7807</a> ), or the "Problem Details for HTTP APIs". This service is used instead of the <a href="#">SetResponseStatus</a> and <a href="#">SetResponseBody</a> .
<b>GetErrorResponse</b>	Creates and returns an error response. This is intended to be used when there is no response body to return to the client. This response will contain the current response status and any error information stored using <a href="#">Error_Services</a> .
<b>DecodePercentString</b>	Decodes the string so that percent-encoded characters are restored to their regular form. This returns the decoded string.
<b>ClearSettings</b>	Clears all of the global common variables used to track header names, values, and the status settings. This will typically only be called within <a href="#">HTTP_MCP</a> when the response is finished and sent back to the OECGI.
<b>GetHomeURL</b>	Returns the Home URL for the web site. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP</a> <1>.

<b>GetAPIRootURL</b>	Returns the API Root URL for the web site. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;2&gt;</a> .
<b>GetCapturePath</b>	Returns the capture path for the request and response content. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;3&gt;</a>
<b>GetEnableAuthenticationFlag</b>	Returns the enable authentication flag setting. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;4&gt;</a> . Note: Only an explicit setting of <i>False</i> (0) will turn this flag off.
<b>GetEnableHTTPBasicAuthenticationFlag</b>	Returns the enable HTTP Basic Authentication flag setting. It pulls this from <a href="#">SYSENV\SRP_HTTP_FRAMEWORK_SETUP &lt;15&gt;</a> . Note: Only an explicit setting of <i>False</i> (0) will turn this flag off.
<b>GetNewPasswordTimeToLive</b>	Returns the length of time (in hours) that new passwords can be valid before needing to be reset. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;16&gt;</a> .
<b>GetOldPasswordTimeToLive</b>	Returns the length of time (in hours) that old passwords can be valid. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;17&gt;</a> .
<b>GetInvalidPasswordLimit</b>	Returns the number of times an invalid password attempt can be made before containment action is taken. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;18&gt;</a> .
<b>GetContainmentAction</b>	Returns the containment action to take if the number of invalid password attempts exceeds the limit. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;19&gt;</a> .
<b>SetServerEnabled</b>	Sets the enabled status of the server.
<b>GetServerEnabled</b>	Returns enabled status of the server.
<b>SetTotalInvalidPasswordAttempts</b>	Sets the total number of invalid password attempts made.
<b>GetTotalInvalidPasswordAttempts</b>	Returns the total number of invalid password attempts made.
<b>GetRealmValue</b>	Returns the realm value, which is used by the WWW-Authenticate response header. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;5&gt;</a> .
<b>GetEntryPointService</b>	Returns the entry point service name. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;6&gt;</a> . Default is <i>entry_point</i> .
<b>GetVersion</b>	A carriage-return/line-feed character will be used to separate the two pieces of information.
<b>GetFullEndPointURL</b>	Returns the full URL for the end point. This should correspond with the URL requested by the client.
<b>GetFlushCacheFlag</b>	Returns the flush cache flag setting. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;7&gt;</a> .
<b>GetNonAuthenticatedPaths</b>	Returns the list of non-authenticated URLs. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;8&gt;</a> .
<b>URLRequiresAuthentication</b>	Returns a boolean flag whether the indicated URL requires authentication or not.
<b>GetWhitelistedIPs</b>	Returns the list of whitelisted IPs. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;14&gt;</a> .
<b>IsIPWhitelisted</b>	Returns a Boolean flag whether the indicated IP is whitelisted.
<b>GetWhitelistedIPsType</b>	Returns the whitelisted IPs type. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;23&gt;</a> . Type will either be a 1 (these IPs will only be permitted) or a 2 (these IPs will always be permitted).

<b>GetBannedIPs</b>	Returns the list of banned IPs. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;22&gt;</a> .
<b>IsIPBanned</b>	Returns a Boolean flag whether the indicated IP is whitelisted.
<b>AddBannedIP</b>	Adds the indicated IP to the list of banned IPs.
<b>RemoveBannedIP</b>	Removes the indicated IP to the list of banned IPs.
<b>IsIPPermitted</b>	Returns a Boolean flag whether the indicated IP is permitted or not. Note, an IP is permitted if the whitelist is empty or if it is included in the list of whitelisted IPs. (This is a synonym service to <a href="#">IPsPermitted</a> .)
<b>IPsPermitted</b>	Returns a Boolean flag whether the indicated IP is permitted or not. Note, an IP is permitted if the whitelist is empty or if it is included in the list of whitelisted IPs. (This is a synonym service to <a href="#">IsIPPermitted</a> .)
<b>GetBestContentNegotiation</b>	Returns the best content negotiation match based on the options the server is able to support and the options the client requested in one of the header request fields that supports content negotiation.
<b>GetAbortedService</b>	Returns the service handler for aborted HTTP Requests. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;9&gt;</a> .
<b>GetEnableLoggingFlag</b>	Returns the enable logging flag setting. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;10&gt;</a> . Note: Only an explicit setting of <i>False</i> (0) will turn this flag off.
<b>GetLogErrorsOnlyFlag</b>	Returns the log errors only flag setting. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;13&gt;</a> . Note: Only an explicit setting of <i>False</i> (0) will turn this flag off.
<b>GetDebuggerSetting</b>	Returns the value that will be passed into the <a href="#">RTI_Debugger_Setting</a> subroutine. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;11&gt;</a> . If no value has been set then a 0 (disabled) is returned as the default.
<b>GetDebuggerService</b>	Returns the service handler for aborted HTTP Requests due to runtime errors. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;12&gt;</a> .
<b>GetAPICallProcedure</b>	Returns the API calling procedure method which is primarily used by <a href="#">HTTP_MCP</a> . It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;20&gt;</a> . If no value has been set then "Web API" is returned as the default.
<b>GetWebAPI</b>	Returns the name of the indicated resource's API procedure if it exists. If it does not exist then an empty string will be returned.
<b>UpdateWebAPIs</b>	Creates or updates exist web API procedures as needed based on the indicated resource list. If at least one new API has been updated, this service will return a <i>True\$</i> . Otherwise a <i>False\$</i> will be returned. Note: this will not remove existing API procedures or individual APIs even if the resource is no longer referenced. The <i>MakeLocal</i> argument can indicate if inherited APIs should be made local in order to be updated and compiled successfully.
<b>GetLocalAppKeyID</b>	Returns the local application KeyID for the indicated row. If one does not exist, the FRAMEWORKS row will be copied.
<b>GetEndpoint</b>	Returns the endpoint for the current request. If this is the entry point then "APIROOT" will be returned so the caller knows explicitly what type of endpoint this is.
<b>GetAPIVersion</b>	Returns the API version being request.
<b>GetCookies</b>	Returns all cookie strings from the request headers using the indicated delimiter. If the Delimiter argument is empty, all cookie strings will be delimited with @FM.
<b>GetCookie</b>	Returns the value for the indicated cookie name.
<b>IsIPWhitelisted</b>	Returns a Boolean flag whether the indicated IP is whitelisted.
<b>AddBannedIP</b>	Adds the indicated IP to the list of banned IPs.
<b>RemoveBannedIP</b>	Removes the indicated IP to the list of banned IPs.
<b>IsIPBanned</b>	Returns a Boolean flag whether the indicated IP is whitelisted.
<b>GetBannedIPs</b>	Returns the list of banned IPs. It pulls this from <a href="#">SRP_HTTP_FRAMEWORK_SETUP &lt;22&gt;</a> .

## Params

The proper use of the generic arguments are defined in the definition of each service above.