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. **Required.**
@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
RunWebAPI	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.
RunHTTPService	Calls the indicated HTTP web service routine.
SetSelfURL	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.
GetSelfURL	Returns the self URL for the current service.
SetSessionID	Creates and sets a unique Session ID for the current HTTP Request/Response. This is used to stamp various logs.
GetSessionID	Returns the unique Session ID for the current HTTP Request/Response. This is used to stamp various logs.
CreateLogFile	Creates a log file in the designated capture path.
SetOECGIRequest	Sets the HTTP request that the OECGI creates to memory so it can be retrieved by later routines. This avoids the need to pass this around into various routines.
GetOECGIRequest	Returns the original HTTP request that the OECGI creates.
SetOECGIProcErr	Sets the HTTP ProcErr that the OECGI creates to memory so it can be retrieved by later routines. This avoids the need to pass this around into various routines.
GetOECGIProcErr	Returns the original HTTP ProcErr that the OECGI creates.
SetHTTPValue	Sets a specific HTTP request value. This is normally set within the *SetOECGIRequest* service and done directly using Memory_Services for efficiency, but some APIs might need to override an HTTP request value.

GetHTTPValue	Returns a specific HTTP request value. This is normally set within the *SetOECGIRequest* 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.
SetRequestHeaderFields	Sets all of the Request Header Fields based on the content of the HTTP request that the OECGI creates. This assumes the *SetOECGIRequest* service has already been called so that the Request array is in memory.
SetRequestHeaderField	Sets the indicated Request Header Field with the indicated value. This can then be retrieved with a *GetRequestHeaderField* service call so that server processing can operate accordingly.
GetRequestHeaderFields	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.
GetRequestHeaderField	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.
SetQueryFields	Sets all of the Query Fields based on the content of the HTTP request that the OECGI creates. This assumes the *SetOECGIRequest* service has already been called so that the Request array is in memory.
SetQueryField	Sets the indicated Query field with the indicated value. This can then be retrieved with a *GetQueryField* service call so that server processing can operate accordingly.
GetQueryField	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.
SetResponseHeaderField	Sets the indicated Response Header Field with the indicated value. This can then be retrieved with a *GetRequestHeaderField* service call so that server processing can operate accordingly.
SetCookie	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.
GetResponseHeaderFields	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.
GetResponseHeaderField	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.
SetResponseStatus	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.
GetResponseStatus	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.
SetResponseBody	Sets the body content to be returned in the response.
GetResponseBody	Gets the current body to be returned in the response.
GetResponseBodyIsBinary	Gets the binary flag associated with the current body. This is normally set by the *SetResponseBody* service at the time the body content is also set.
GetResponse	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.
SetResponseError	Sets a response error. This follows the RFC 7807 specification (https://tools.ietf.org/html/rfc7807), or the "Problem Details for HTTP APIs". This service is used instead of the *SetResponseStatus* and *SetResponseBody*.
GetErrorResponse	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 Error_Services.
DecodePercentString	Decodes the string so that percent-encoded characters are restored to their regular form. This returns the decoded string.
ClearSettings	Clears all of the global common variables used to track header names, values, and the status settings. This will typically only be called within HTTP_MCP when the response is finished and sent back to the OECGI.
GetHomeURL	Returns the Home URL for the web site. It pulls this from SRP_HTTP_FRAMEWORK_SETUP <1>.

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

GetBannedIPs	Returns the list of banned IPs. It pulls this from SRP_HTTP_FRAMEWORK_SETUP <22>.
IsIPBanned	Returns a Boolean flag whether the indicated IP is whitelisted.
AddBannedIP	Adds the indicated IP to the list of banned IPs.
RemoveBannedIP	Removes the indicated IP to the list of banned IPs.
IsIPPermitted	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 IPIsPermitted.)
IPIsPermitted	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 IsIPPermitted.)
GetBestContentNegotiation	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.
GetAbortedService	Returns the service handler for aborted HTTP Requests. It pulls this from SRP_HTTP_FRAMEWORK_SETUP <9>.
GetEnableLoggingFlag	Returns the enable logging flag setting. It pulls this from SRP_HTTP_FRAMEWORK_SETUP <10>. Note: Only an explicit setting of *False* (0) will turn this flag off.
GetLogErrorsOnlyFlag	Returns the log errors only flag setting. It pulls this from SRP_HTTP_FRAMEWORK_SETUP <13>. Note: Only an explicit setting of False (0) will turn this flag off.
GetDebuggerSetting	Returns the value that will be passed into the RTI_Debugger_Setting subroutine. It pulls this from SRP_HTTP_FRAMEWORK_SETUP <11>. If no value has been set then a 0 (disabled) is returned as the default.
GetDebuggerService	Returns the service handler for aborted HTTP Requests due to runtime errors. It pulls this from SRP_HTTP_FRAMEWORK_SETUP <12>.
GetAPICallProcedure	Returns the API calling procedure method which is primarily used by HTTP_MCP. It pulls this from SRP_HTTP_FRAMEWORK_SETUP <20>. If no value has been set then "Web API" is returned as the default.
GetWebAPI	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.
UpdateWebAPIs	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 True$. Otherwise a False$ will be returned. Note: this will not remove existing API procedures or individual APIs even if the resource is no longer referenced. The *MakeLocal* argument can indicate if inherited APIs should be made local in order to be updated and compiled successfully.
GetLocalAppKeyID	Returns the local application KeyID for the indicated row. If one does not exist, the FRAMEWORKS row will be copied.
GetEndpoint	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.
GetAPIVersion	Returns the API version being request.
GetCookies	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.
GetCookie	Returns the value for the indicated cookie name.
IsIPWhitelisted	Returns a Boolean flag whether the indicated IP is whitelisted.
AddBannedIP	Adds the indicated IP to the list of banned IPs.
RemoveBannedIP	Removes the indicated IP to the list of banned IPs.
IsIPBanned	Returns a Boolean flag whether the indicated IP is whitelisted.
GetBannedIPs	Returns the list of banned IPs. It pulls this from SRP_HTTP_FRAMEWORK_SETUP <22>.

Params

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