What's New

- 4.0.10
  - OECGI 4.0.2 Love
  - Baking More Cookies
  - What About Blob?
- 4.0.9
  - IP Banning
- 4.0.8
  - OECGI4.php Love
- 4.0.7
  - C is for Cookie, That's Good Enough For Me
- 4.0.6
  - FrameWorks Friendlier
  - Managing Resources a Little Easier
  - Don't lose your HEAD
  - URL Templates and HAL
  - Embracing SRP Utilities 2.1
- 4.0.5
  - More Logging TLC
  - Image Automation
  - Here's a Wild Idea...
  - It's a Bug Hunt
- 4.0.4
  - Breaking the Log Jam
  - Paving the Road to Integration
- 4.0.3
  - Ping Me Up Sometime
  - To Log, or Not to Log, that is the Question
  - Home URL on the Range
  - Time Off for Good Behavior
- 4.0.2
  - A Good Thing just gets Better
- 4.0.1
  - A Good Thing just gets Better
- 4.0.0
  - API Automation
  - Authentication in a Box
  - APIs are the new Services
  - Resource Building Your Way
- 3.0.4
  - Protecting Column Data
- 3.0.3
  - Keeping in Step
- 3.0.2
  - Missing Metadata
- 3.0.1
  - HTTP Framework Setup
  - Log like a Boss
  - Log File Names
  - Making Contacts
  - Ah CRUD...
- 3.0.0
  - NDW_HTTP_FRAMEWORK_SETUP
  - Non-Authenticated URLs
  - HTTP_MCP and Logging

4.0.10

2023-09-04

OECGI 4.0.2 Love

In older versions of the OECGI, if a developer wanted to capture the values of request headers that were not automatically included in the Request array, a comma delimited list of these headers needed to be entered into the Registry AdditionalValues string. As of OECGI 4.0.2, an asterisk character (*) can be entered instead. This puts all request headers into the Request array. All relevant services within the HTTP_Services module have been updated to be backwards and forward compatible.

Baking More Cookies

In v4.0.7 we provided a *GetCookies* and a *GetCookie* service in the HTTPClient_Services module. This was all well and good when handling API requests to other servers, but what about those cookies that were returned to your own APIs? Worry no more because we now have a *GetCookies* and *GetCookie* service in the HTTP_Services module to make it much easier to eat those tasty morsels.

What About Blob?

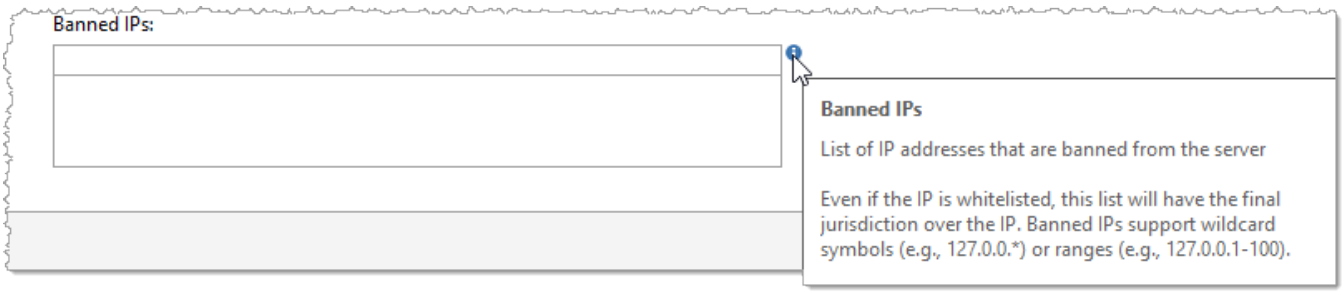
Sending a binary object to another server via a HTTP request doesn't always work. We have added a new argument, *ResponseIsBinary*, to the *SendHTTP Request* service (a member of the HTTPClient_Services module) to force the a "blob" datatype. Note, this requires SRP Utilities 2.2.6 or higher.

4.0.9

2020-07-08

IP Banning

As a compliment to the existing Whitelisted IPs feature we now offer an option to Banned IPs:



4.0.8

2020-03-04

OECGI4.php Love

Perhaps you don't know this, but Revelation Software provides two plug-ins for HTTP connectivity. The original and most likely used is OECGI4.exe (or OECGI3.exe for older systems). This is obviously designed for Windows web servers. The other version is OECGI4.php (and there is also an OECGI3.php). This version would be the plug-in of choice for non-Windows (usually Linux) based web servers. We installed the PHP version several years ago on a few sites and experienced no issues with it. However, a customer recently brought to our attention that their HTTP responses were coming back with the headers appearing with the body. Not good! Upon review and with some insight from the ever helpful Bryan Shumsky, we discovered that the PHP plug-in expects the response headers to be separated from the body with LF as the delimiter. The HTTP Framework has been using CR/LF as the delimiter all this time. There is a reason we use CR/LF delimiters...because this is what the HTTP specification requires. What we didn't know was that the PHP plug-in parses our response (expecting LF delimiters) and then builds a property response using PHP functions. Since we were using CR/LF delimiters, the parsing logic failed and kept the header and the body together. This isn't an issue for OECGI4.exe because it basically passes through the response "as is". Did you get all that? **TL;DR**: The HTTP Framework now checks to see if the EXE or PHP plug-in is being used and sends back the response using the correct delimiters.

4.0.7

2020-02-27

C is for Cookie, That's Good Enough For Me

Cookies have long been a valuable tool for maintaining state, implementing authentication, and tracking history with web applications. The SRP HTTP Framework has always supported cookies via the *SetResponseHeaderField* service (a member of the HTTP_Services module) and the *GetResponseHeaderField* service (a member of the HTTPClient_Services module). However, a recent project convinced us that Cookies needed a little first-class citizen love. Therefore we have added a few new services so baking and consuming cookies is much yummier:

- ***SetCookie*** (a member of the HTTP_Services module) - 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.
- ***GetCookies*** (a member of the HTTPClient_Services module) - Returns all cookie strings from the response headers.
- ***GetCookie*** (a member of the HTTPClient_Services module) - Returns the value for the indicated cookie name.

4.0.6

2020-02-04

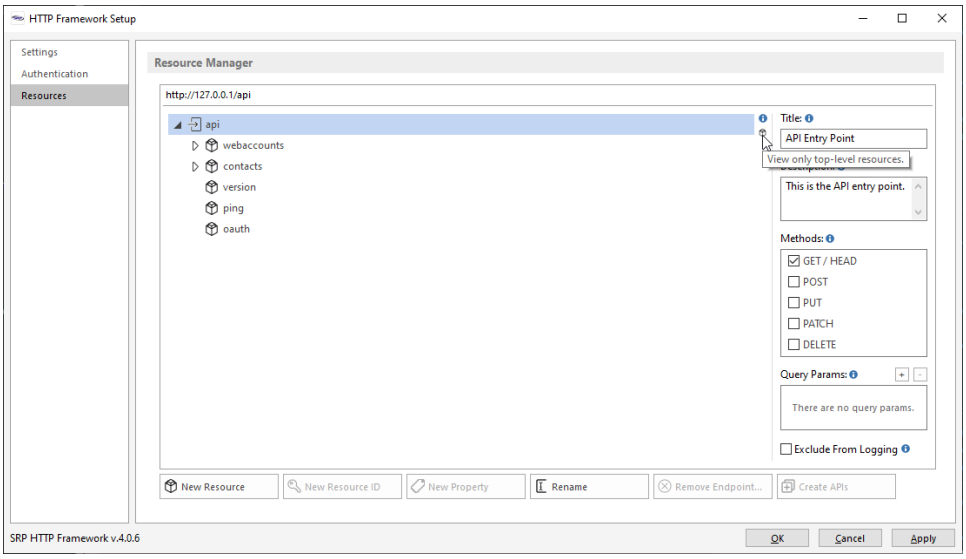
FrameWorks Friendlier

The SRP HTTP Framework has always been included with the full SRP FrameWorks package. However, the core forms (e.g., *HTTP Framework Setup*, *Web Accounts*, and *HTTP Logs*) were never designed with the intent of running in parallel with the main SRP FrameWorks MDI Frame. A client reported to us that after running our SRP HTTP Framework forms, their main MDI Frame stopped responding to clicks in the Ribbon control. We identified the problem being related to a memory cache conflict. SRP HTTP Framework now uses its own cache and so the two systems are playing nicely with each other again.

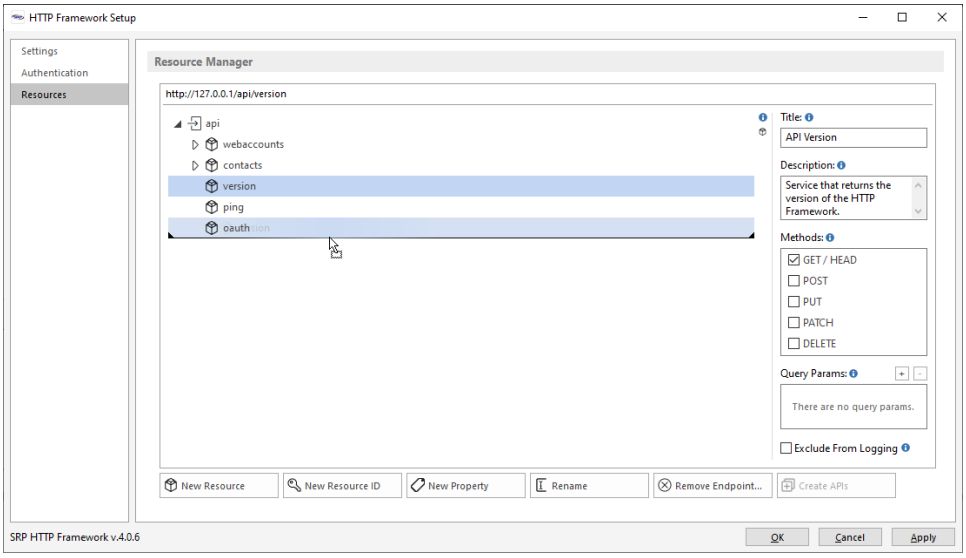
Managing Resources a Little Easier

The *Resource Manager* page of the *HTTP Framework Setup* form has a couple of new quality of life features:

- **Resource Collapse.** The hierarchical resource view can tend to get very busy when multiple resources are open. There are times when we just want to collapse everything to avoid excessive scrolling up and down. A new (fairly inconspicuous) button has been added to do just that:



- **Resource Sorting.** When new resources are added, they always land at the bottom. However, there are times when we simply want to view our resources in a different order than when their creation date. Now this can be accomplished with a simple drag-and-drop action (shame on us for not adding this sooner)!



Don't lose your HEAD

HEAD is one of the first methods introduced in the HTTP spec, but it is probably one of the lesser known members. According to the specification, HEAD works exactly like GET but it does not return a body. So, why would anyone use it? First, it is a way to "test the waters" of a GET request without the overhead that this request might produce. Second, HEAD requests are supposed to return the same HTTP Response headers that a GET request, making it easier to anticipate how the body will be returned (e.g., the *Content-Type* header should still have a valid value, even if the body is empty). Finally, since the HEAD method requires authentication, it is a great way to "log into" an application without requiring the app to download and endpoint's resource right away. So it seemed time that the SRP HTTP Framework finally give some respect to this often overlooked method.

You may have noticed in the previous screen shots that the GET method displays the HEAD method next to it. This is to indicate that GET and HEAD methods will be enabled or disabled together. Furthermore, all API routines will automatically get a HEAD API signature added above the GET API signature:

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////
// Endpoint Handlers
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////

API contacts.HEAD
API contacts.GET

    Method  = HTTP_Services('GetHTTPRequestMethod')
        // If the GET method then return a body. If the HEAD method then just return response headers.
    .
    .
    .
    .

end api
```

URL Templates and HAL

URI Templates (URL Templates going forward) were introduced in 2012 as a way of documenting the structure of a URL (including any query params). One primary purpose for URL Templates is for "the discovery of available services". This fits very well with the REST constraint of "resource identification". Simply put, a URL Template is an abstraction (or generic version) of a resource URL. A templated version of a URL contains arguments surrounded by curly braces like so:

`https://www.examples.org/api/contacts/{contactID}`

The HAL specification supports links to URL Templates (versus resource links). To help support URL Templates, the *AddLinkRelation* service (a member of HTTP_Resource_Services) has been updated with new arguments to make it easy to include a URL Template. Any valid resource URL within the system can be passed into the *URL* argument and if the *IsTemplate* argument is set to 1, then it will be converted to a URL Template. For example, the SRP HTTP Framework automatically ships with a Contacts resource which can be accessed via this URL:

`https://www.examples.org/api/contacts`

There are three query params supported by this resource: *first_name*, *last_name*, and *state*. If the above URL is passed into the *AddLinkRelation* service with the *IsTemplate* argument set to 1, then resulting URL will look like this:

`https://www.examples.org/api/contacts{?first_name,last_name,state}`

The act of converting a URL Template into a valid resource URL is called "expansion". This is something a consumer of your services might consider building. However, at a minimum, any human agent can review the URL Templates you provide and benefit from self-documenting APIs...which is a major objective of REST!

Embracing SRP Utilities 2.1

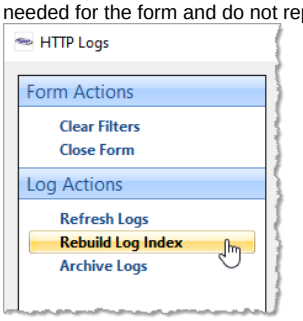
The most recent release of SRP Utilities includes many handy services. The SRP_DateTime function, in particular, has been incorporated into the SRP HTTP Framework. We are using it to replace all calls to Utility_DotNet("TIMEZONE") as a way of producing a UTC date/time stamp. We discovered that in some cases Utility_DotNet is unable to return the proper local time which results in a Get_Status() error...which then leads to 502 Bad Gateway errors. Ouch! Suffice to say that the SRP HTTP Framework 4.0.6 requires SRP Utilities 2.1.

4.0.5

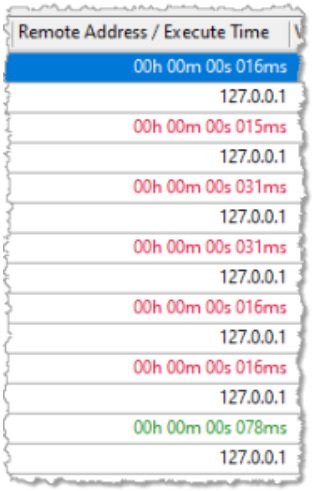
More Logging TLC

The more logging gets used the more ways we see how this can be improved. Here is a summary of what we've added:

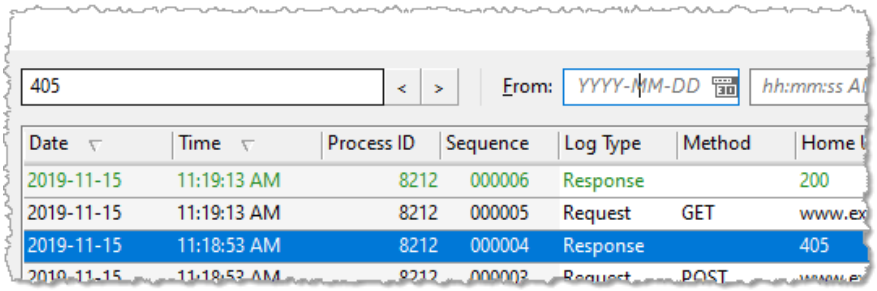
- To help speed up the population of the **HTTP Logs** utility, special log index files are automatically created. These files contain all of the data needed for the form and do not replace the individual log files. You can rebuild your log indexes at anytime.



- The **HTTP Logs** utility has a new column which displays the *Remote Address* (for requests) and the *Execute Time* (for responses).



- Searching logs no longer shrinks the list in the **HTTP Logs** utility but finds the next log matching the search criteria. Users can go forwards and backwards via newly added VCR style *Find Next* and *Find Previous* buttons.



- Logs before and for a specified date can now be archived via the *Archive Logs* menu. This feature zips up your logs and stores them in the *archive* sub-folder.

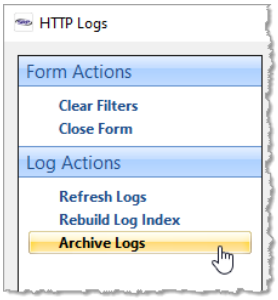
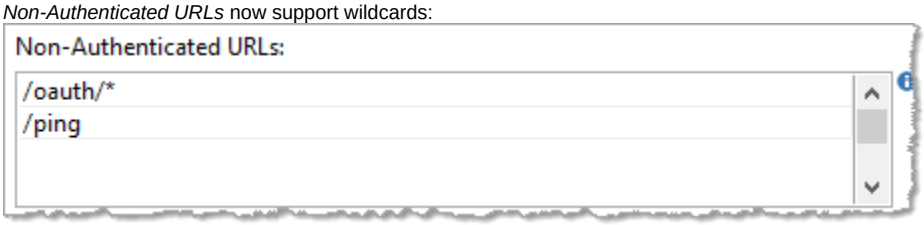


Image Automation

You might have already noticed that setting the **Response** body will often automatically set the *Content-Type* header field to the correct value (e.g., application/xml, application/json, text/html, etc.). This exists as a safety check for when developers forget to specify this. We've updated this feature to properly identify PNG, JPG, and GIF data formats.

Here's a Wild Idea...

- *Non-Authenticated URLs* now support wildcards:



In the above example, an API developer can now support multiple OAuth redirect URLs (e.g., /oauth/google, /oauth/facebook, /oauth/microsoft, etc.) without having to list each one individually.

- *Whitelisted IPs* also support wildcards:



In the above example, an API developer can now support a range of IPs rather than having to list each one individually.

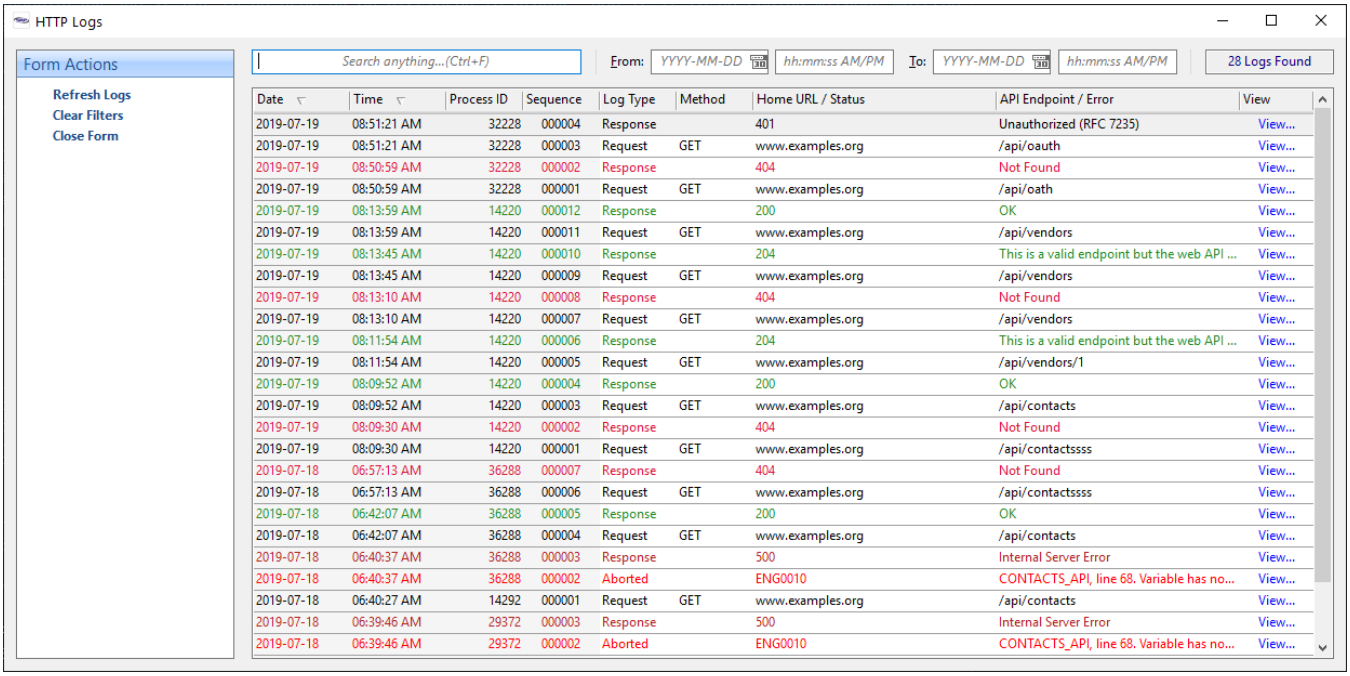
It's a Bug Hunt

As always, we continue to squash bugs that come up in real world scenarios. Private William Hudson would have been proud.

4.0.4

Breaking the Log Jam

In our last release we added a feature so that individual endpoints can be excluded from logging. Still, the number of log files that get generated can still be daunting and difficult to sift through when specific issues need to be investigated. Never fear...the new **HTTP Logs** utility is here! Just launch the **NDW_HTTP_LOGS** form, confirm the log directory, and wonder why we never created this sooner!



Paving the Road to Integration

Up until now, the *Resource Manager* feature of the *HTTP Framework Setup* form maintained exclusive control over resource endpoint definitions (read: *tigh tly coupled code design*). Recent projects necessitated a way for 3rd party utilities to tap into this same functionality. Rather than copy blocks of code, we decoupled the resource endpoint logic and relocated it into its own service module: **HTTP_Resource_Manager_Services**. This will be documented soon but, unless you have a need to manage endpoints outside of the HTTP Framework Setup form, you probably don't need to know about this.

4.0.3

Ping Me Up Sometime

A new **Ping** API has been added. Many systems need a way to monitor whether or not the API server is still running. The Ping API was created specifically for this purpose. It returns the current date and time in GMT format.

To Log, or Not to Log, that is the Question

Logging is great until you need to sort through thousands of logs that exist only to monitor the API server. Each resource endpoint now has the ability to be excluded from logging. The new Ping API is excluded by default.

Home URL on the Range

The HTTP Framework Setup provides a way to enter the expected **Home URL** for your APIs. This is used for logging and for producing hypermedia links in your API responses. However, this might not be useful for sites that support multiple URLs or IP addresses. Therefore, the **Home URL** will now be derived from the incoming request.

Time Off for Good Behavior

The new **Web Accounts** authentication system works really well at tracking the number of invalid password attempts. In fact, it works a little *too* well. It turns out that once an account has incurred an invalid password attempt, this incursion is remembered by the system even after the account successfully provides a valid password. This eventually causes an account to immediately become disabled after only one invalid password attempt. The system will now reset the number of invalid password attempts whenever a valid password is provided. The number of invalid passwords tracked by the overall system, however, is still accumulated until manually reset.

4.0.2

A Good Thing just gets Better

We hate to sound repetitive, but this is again a minor release that ties up more loose ends.

4.0.1

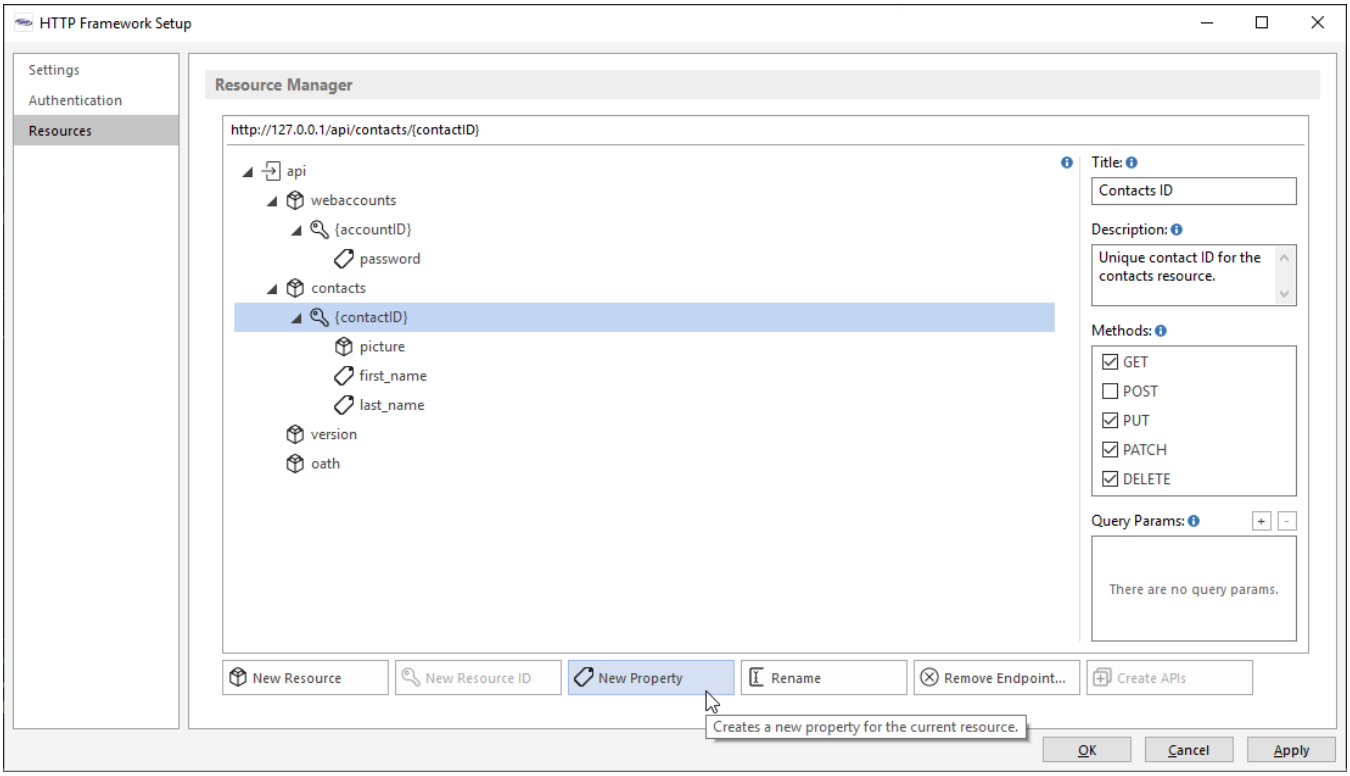
A Good Thing just gets Better

This release mostly ties up a few loose ends that were introduced in v4.0.0.

4.0.0

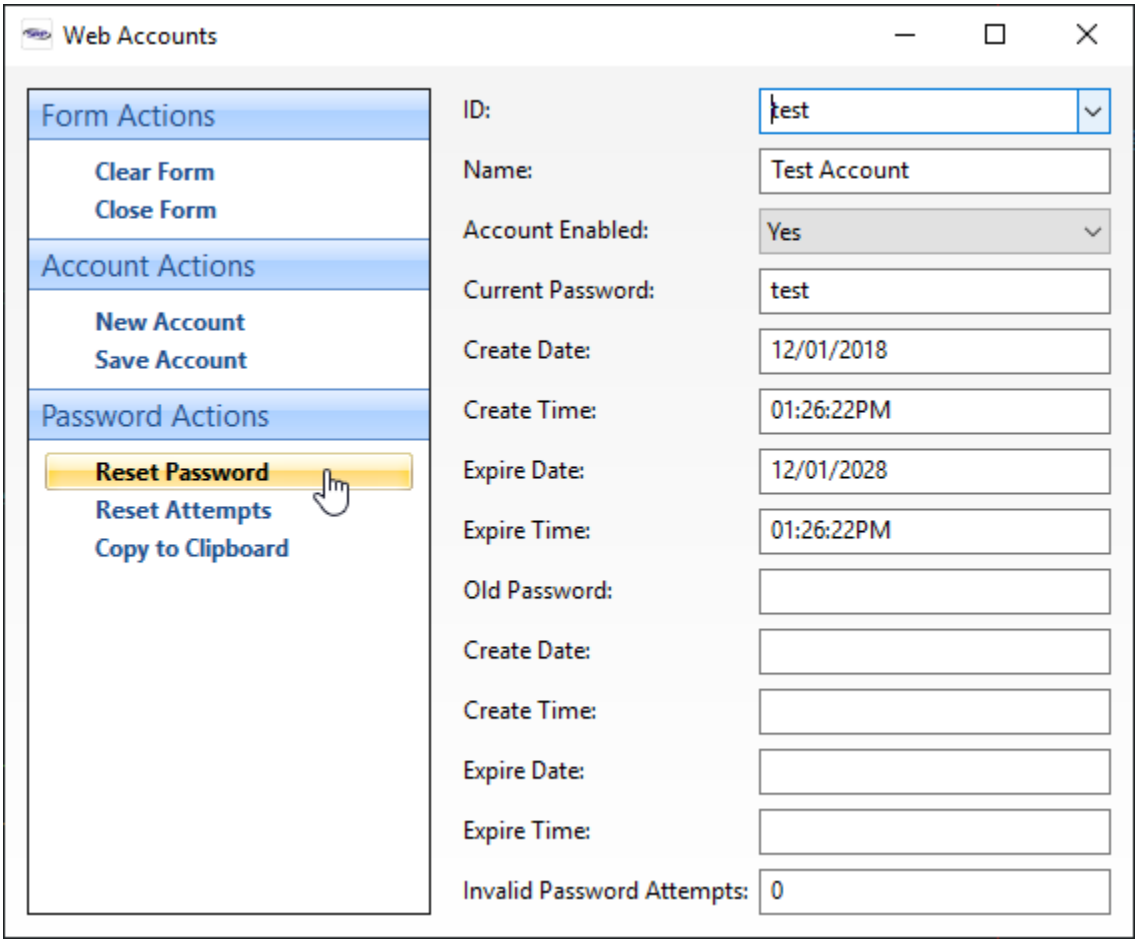
API Automation

Our next major release to the SRP HTTP Framework introduces some major improvements to the way resources and APIs are created. The HTTP Framework Setup form has been heavily extended to support new features. The most significant of these is the **Resource Manager**, a visual tool that allows developers to define endpoints and automatically generate API templates with the click of a few buttons!



Authentication in a Box

While the SRP HTTP Framework has included support for HTTP Basic Authentication since its first release and it has made it easy to embrace other methods of authentication, it had never ship with an out-of-the box authentication management tool. We have rectified this by including a **Web Accounts** manager. Users can now be created with expiring passwords. The SRP HTTP Framework authentication service has been significantly updated to provide support for Web Accounts while making it even easier to support other methods as well. Also, unsuccessful authentication requests are now tracked and the system can be configured to automatically go into containment mode based on a predetermined number of invalid passwords.



APIs are the new Services

As a way of embracing the awesomeness and convenience of our new enhanced BASIC+ syntax, we've added a new statement: **GoToAPI**. This effectively eliminates maintaining those strings of allowed methods and services in addition to the long and complicated Case statements to route each method and endpoint combination to an appropriate GoSub label. Seeing is believing...so check it out!

```
GoToAPI else
   // The specific resource endpoint doesn't have a API handler yet.
   HTTP_Services('SetResponseStatus', 204, 'This is a valid endpoint but a web API handler has not yet been
created.')
end

Return Response OR ''

////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////
// Endpoint Handlers
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////

API contacts.POST
   HTTP_Resource_Services('PostDatabaseItem', 'CONTACTS', FullEndpointURL)
end api

API contacts.ID.PATCH
   KeyID   = EndpointSegment
   HTTP_Resource_Services('PatchDatabaseItem', 'CONTACTS', FullEndpointURL, KeyID)
end api
```

```
API contacts.ID.PUT
   KeyID   = EndpointSegment
   HTTP_Resource_Services('PutDatabaseItem', 'CONTACTS', FullEndpointURL, KeyID)
end api

API contacts.ID.DELETE
   KeyID   = EndpointSegment
   HTTP_Resource_Services('DeleteDatabaseItem', 'CONTACTS', KeyID)
end api
```

Due to the new syntax and the desire to maintain backwards compatibility with the legacy service code, the old format of **HTTP_*{Resource}*_Services** is now replaced with ***{Resource}*_API**.

Resource Building Your Way

The HTTP_Resource_Services routine has always been a convenient way to quickly convert OpenInsight data into rich JSON while following good RESTful practices such as including self-referencing and collection URIs. This is accomplished through a family of high-level services (e.g., GetDatabaseItem, PostDatabaseItem, DeleteDatabaseItem). However, there are times when a little more low-level effort is required. To help with this, several new services have been added to be a middle ground between these high-level services and direct calls to the SRP_JSON function.

3.0.4

Protecting Column Data

In addition to a couple of minor bug fixes, this release enhances POST, PUT, and PATCH APIs that rely upon HTTP_Resource_Services. Developers can now pass in an @FM delimited list of column names that are allowed to be updated. Hence, even if the client submits a payload containing every known column name in the dictionary, these services will now cross-check each column name against the approved list and by-pass any columns that are not mentioned. If this *allowed column names* list is empty, then these APIs will attempt to update every column referenced in the payload (i.e., default behavior).

3.0.3

Keeping in Step

Another minor release to allow the *Entry Point Service* entry to be formatted like the other service routines in the HTTP Framework Setup form. This prompt normally expects "HTTP" and "SERVICES" to be omitted, but the more recent *Aborted Service* and *Debugger Setting* entries expect fully qualified names. With this update, **entry_point** or **HTTP_ENTRY_POINT_SERVICES** will work equally well.

3.0.2

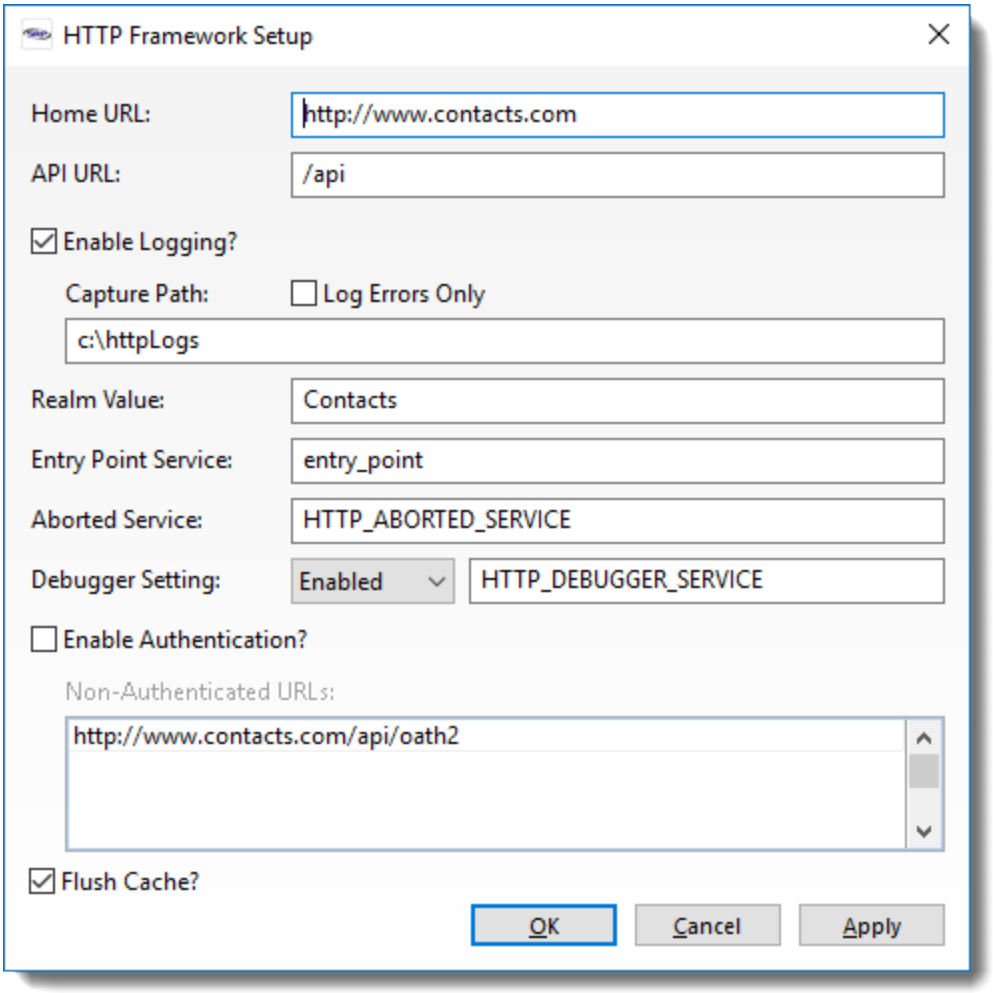
Missing Metadata

What!? A new version already? Yes, but nothing to get too excited over. We realized that even though most of our routines are written using a service-oriented approach, we were not shipping the metadata that the SRP Editor can use to help automate the building of calls into our services. This oversight is now rectified.

3.0.1

In version 3.0.1, virtually every major component has been overhauled to make RESTful web services even easier to create and debug. Just read on...

HTTP Framework Setup



Introduced in v3.0.0, the **HTTP Framework Setup** form (*NDW_HTTP_FRAMEWORK_SETUP**)** form includes more options:

- **Enable Logging** - No longer is it necessary to rename your log folder to turn off logging. Now you can simply toggle this on or off!
- **Log Errors Only** - Are you getting lost in a sea of log files? Perhaps you only need to find those pesky error responses. Easy peasy! Just check this box and only 4xx and 5xx HTTP status code responses will be sent to your folder.
- **Aborted Service** - Miss having an INET_ABORTED equivalent feature? You're in luck. A template for handling aborted requests now ships with v3.0.1 called HTTP_ABORTED_SERVICE. Feel free to modify this or copy it into your own custom aborted service handler.
- **Debugger Setting** - Tired of running back and forth to the Database Manager to change the Debugger Setting? Are you concerned that modifying this setting will affect your desktop users? If you answered yes to either question we have you covered! Now you can specify your own Debugger Setting that will only affect your web services. Of course, what good is having an *Intercept* option without a *Debugger Intercept* service? Glad you asked because we have that covered too. The HTTP_DEBUGGER_SERVICE routine also ships with v3.0.1. Like HTTP_ABORTED_SERVICE, you are free to modify this routine or make a copy to use.

Log like a Boss

Logging is a critical tool in the developer's belt when troubleshooting and profiling web services. In the last release we updated the Response log to include useful metadata. In 3.0.1, we go full throttle with the logging enhancements.

- **Request Logs** - Once upon a time the Request logs were nothing but raw text dumps of the *Request* argument that the OECGI passed into HTTP_MCP. We are now proud to showcase our new and improved Request log format:

Request Log

```
Request Argument
-------------------------------------------------------------------------------
<01> HTTPQueryString          :
<02> HTTPPathInfo             : contacts
<03> HTTPContentType          :
<04> HTTPContentLength        : 0
<05> HTTPGatewayInterface     : CGI/1.1
<06> HTTPHTTPS                : off
<07> HTTPAccept               : */*
```

```
<08> HTTPCookie               :
<09> HTTPFrom                 :
<10> HTTPReferer              :
<11> HTTPUserAgent            : PostmanRuntime/6.1.6
<12> HTTPTranslated           : C:\MyWebsite\www\contacts
<13> HTTPRemoteAddr           : 127.0.0.1
<14> HTTPRemoteHost           :
<15> HTTPRemoteIdent          :
<16> HTTPRemoteUser           :
<17> HTTPRequestMethod        : GET
<18> HTTPScriptName           : /cgi-bin/oecgi4.exe
<19> HTTPServerName           : www.contacts.com
<20> HTTPServerPort           : 80
<21> HTTPServerProtocol       : HTTP/1.1
<22> HTTPServerSoftware       : Abyss/2.9.3.6-X1-Win32 AbyssLib/2.9.3.6
<23> HTTPServerURL            :
<24> HTTPNoURLDecode          :
<25> HTTPResponseIsBinary     :
<26> HTTPRegistrySettings     +
     RegistryInfo             : SOFTWARE\RevSoft\OECGI4
     EngineName               :
     ServerURL                : localhost
     ServerPort               : 8088
     ApplicationName          : FRAMEWORKS
     UserName                 : FRAMEWORKS
     StartupFlags             : 1
     ShutdownFlags            : 1
     FilePath                 :
     FilePathMapped           :
     FileMode                 : 1
     SysDownPage              :
     OILocation               :
     AdditionalValues         : HTTP_MEDIA_TYPE,HTTP_ACCEPT_ENCODING,HTTP_ACCEPT_CHARSET,
HTTP_ACCEPT_LANGUAGE,HTTP_AUTHORIZATION
<27> HTTPOECGIVersion         : VERSION:OECGI4
<28> HTTPGetString            :
<29> HTTPPostString           :
<30> HTTPAdditionalValues     +
     Media-Type               :
     Accept-Encoding          : gzip, deflate
     Accept-Charset           :
     Accept-Language          :
     Authorization            :
```

- **Response Logs** - Turns out more metadata is a good thing so we went ahead and updated the Response log with the current *HTTP Framework version*, *Authorization information*, and *Query Params* used:

Response Log

```
HTTP Framework : v3.0.1 - 07/11/2017 12:58PM
Time to Execute: 00h 00m 00s 063ms
Request Method : GET
Request URL    : http://www.contacts.com/api/contacts
Authorization  : None
Query Params   : company=benton
--------------------------------------------------------------------------------
Status: 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/hal+json
Content-Length: 507
{
   "_embedded" : {
      "item" : [
         {
            "_links" : {
               "self" : {
                  "href" : "http://www.contacts.com/api/contacts/1"
               }
            },
```

```
            "company" : "Benton, John B Jr",
            "email" : "jbutt@gmail.com",
            "first_name" : "James",
            "last_name" : "Butt"
         }
      ]
   },
   "_links" : {
      "self" : {
         "href" : "http://www.contacts.com/api/contacts"
      }
   }
}
```

- **Aborted Logs** - We added a new log type for aborted web services to make it easy to identify the offending code:

Aborted Log

```
ProcErr Argument
--------------------------------------------------------------------------------
ENG0010:  HTTP_CONTACTS_SERVICES, line 315.  Variable has not been assigned a value.
```

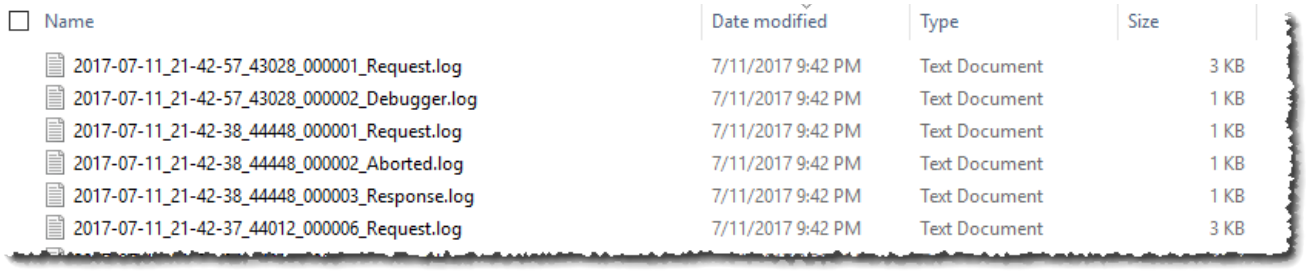
- **Debugger Logs** - For those who want to utilize a Debugger Intercept option for more indepth troubleshooting in their web applications, we also provide a JSON formatted log file for the basic intercept content:

Debugger Log

```
{
   "CallDepth" : 5,
   "CallStack" : [
      {
         "LineNo" : 315,
         "ProcName" : "HTTP_CONTACTS_SERVICES"
      },
      {
         "LineNo" : 155,
         "ProcName" : "HTTP_SERVICES"
      },
      {
         "LineNo" : 81,
         "ProcName" : "HTTP_ENTRY_POINT_SERVICES"
      },
      {
         "LineNo" : 155,
         "ProcName" : "HTTP_SERVICES"
      },
      {
         "LineNo" : 135,
         "ProcName" : "HTTP_MCP"
      }
   ],
   "Curr_Program" : "HTTP_CONTACTS_SERVICES",
   "LineNo" : 315,
   "SPStatCode" : "ENG0010:  HTTP_CONTACTS_SERVICES, line 315.  Variable has not been assigned a value.",
   "SPStatus" : 1
}
```

Log File Names

What's in a name? Actually, quite a bit. We've even updated the names assigned to our log files for more diagnostic muscle:



Name	Date modified	Type	Size
2017-07-11_21-42-57_43028_000001_Request.log	7/11/2017 9:42 PM	Text Document	3 KB
2017-07-11_21-42-57_43028_000002_Debugger.log	7/11/2017 9:42 PM	Text Document	1 KB
2017-07-11_21-42-38_44448_000001_Request.log	7/11/2017 9:42 PM	Text Document	1 KB
2017-07-11_21-42-38_44448_000002_Aborted.log	7/11/2017 9:42 PM	Text Document	1 KB
2017-07-11_21-42-38_44448_000003_Response.log	7/11/2017 9:42 PM	Text Document	1 KB
2017-07-11_21-42-37_44012_000006_Request.log	7/11/2017 9:42 PM	Text Document	3 KB

The format of each file name is `YYYY-MM-DD_HH-MM-SS_PID_LogType.log`. The PID (*Process ID*) can be quite useful when multiple requests are processed simultaneously. Each OpenEngine process will have its own unique PID. Since each HTTP Request and associated Response will be handled by the same OpenEngine, it's quite easy now to pair up the relevant log files. The PID also makes it easy to see when one OpenEngine process may have been terminated due to a runtime error. For instance, in the above screen shot the Debugger log file introduces a new OpenEngine (PID 43028) because the original OpenEngine (44448) used by the previous HTTP Request terminated.

Making Contacts

No, we are not adding social networking to the product but we are offering a full featured Contacts web service. A sample HTTP_CONTACTS_SERVICES has shipped with the product for some time, but it was mostly there as example code and was never connected to a database table. No more! We are now including a sample CONTACTS database table (tied to the GLOBAL database). Just attach and then run the Contacts web service out of the box. We have also updated HTTP_CONTACTS_SERVICES to support all major CRUD activities, including searching via query parameters. We've added a lot of comments to this routine to get both novice and experienced web developers productive even quicker. Use this routine as a template for your own web services or copy bits and pieces to augment exist services.

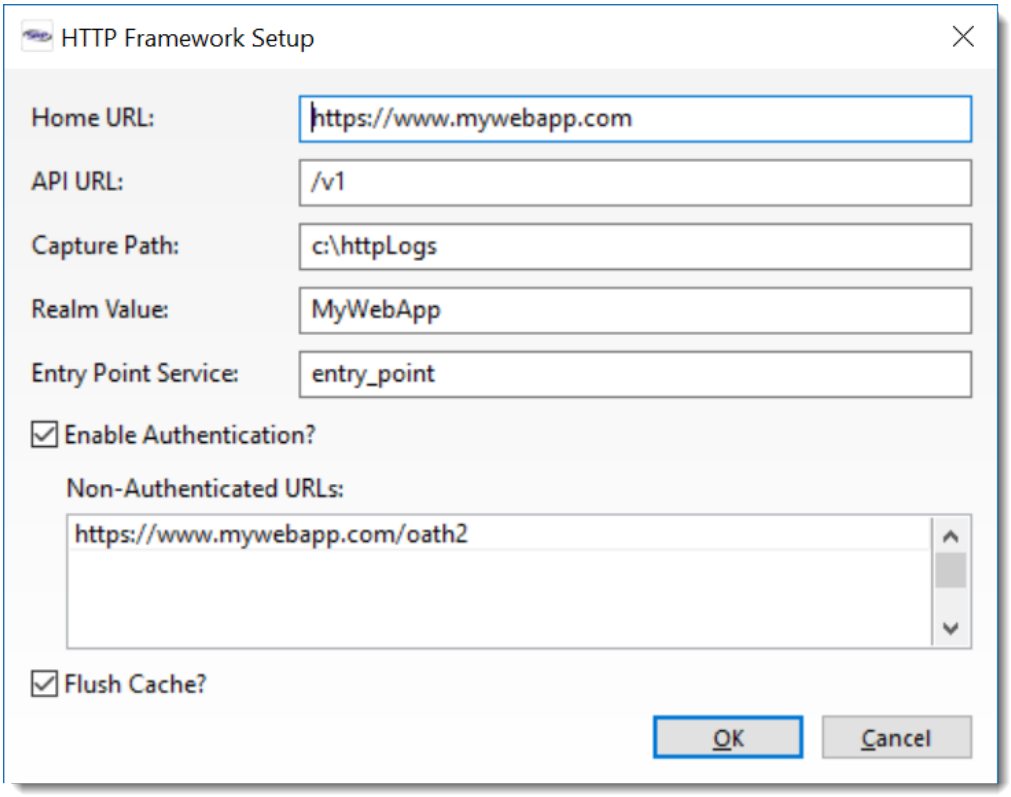
Ah CRUD...

Speaking of *create*, *read*, *update*, and *delete* functionality, we enhanced the HTTP_RESOURCE_SERVICES utility to fully support proper POST, PUT, and PATCH operations against database table resources. POST will create new database rows when the server needs to generate the next Key ID. PUT creates (or updates) database rows when the client specifies the Key ID. Finally, PATCH updates only those specific columns that are passed into the web service (*OECGI4 v4.0.0.3 or higher required*).

3.0.0

Version 3.0.0 introduces new features to make it easier to configure setup values and to accept special URLs without requiring authentication.

NDW_HTTP_FRAMEWORK_SETUP



This form can be used to configure all setup parameters without the need to edit the SRP_HTTP_FRAMEWORK_SETUP record directly.

Non-Authenticated URLs

A new setup parameter has been created to store one or more URLs that should pass through the authentication logic. The HTTP_Authentication_Services module has been updated to verify if the URL requires authentication. If not, the current URL request is marked as authenticated so it can continue to be processed by the end point API.

This feature is helpful when OAuth-type requests need to be supported. OAuth relies upon communication between trusted servers, away from client-side web and mobile applications. This is part of the security mechanism of OAuth. Generally, these URLs cannot be configured to support server-side authentication so these need to be allowed through without authentication.

HTTP_MCP and Logging

The logging feature of HTTP_MCP has been updated to provide more useful information in the response log. In addition to the basic response information (e.g., status code, response headers, and the body), metadata for the URL request itself is added so that it is easier to match the request to the response. Additionally, the time to execute the request from start to finish is also included as a way of helping developers identify potential bottlenecks. Here is an example of an updated response log:

Response Log

```
Time to Execute: 00h 00m 00s 875ms
Request Method : POST
Request URL    : https://api.srpcs.com/srpteam/users/don
--------------------------------------------------------------------------------
Status: 201 Created
Access-Control-Allow-Origin: *
Content-Type: application/hal+json
Content-Length: 129
{
   "URL" : "https://api.srpcs.com/srpteam/users/don",
   "method" : "POST",
   "phrase" : "Created",
   "status" : 201
}
```