http://remedylegacy.com

APL Restful ARDBC Plugin v4.13.4

Contents

Introduction..... .1
Getting Started..... .2
Configuring the Plugin..... .3
Configuring New Restful Service..... .5
Table Record..... .5
Field Records..... .8
Vendor Form Creation..... .12
Consumption of Remote Services..... .15
Templating..... .15
Token Authentication..... .16
Debugging and Tweaking..... .16
Release Notes..... .18
Appendix A: Determining Record Indicator Element/Level..... .31
Appendix B: Skipping JSON Levels..... .33

Introduction

This is an ARDBC program, that means that it allows you to create Vendor forms that the plugin controls. A call to the Vendor form enacts the plugin capabilities. This ARDBC plugin enables consumption of Restful Services via Remedy workflow. Your workflow interacts with the Vendor form through Active Links, Filters, Escalations, Table fields, etc. You can treat the Vendor form in the same manner you would any other Regular form, but instead of the data existing inside of Remedy, it exists in the remote system and you are interacting with it through Rest

This plugin is open-source. I support this plugin through email and phone (at my discretion), you are welcome to take this functional framework that either use it OOTB or as a kickoff point to enhance it to handle your unique needs that I wasn't able to anticipate while writing it. I welcome you to contact me and request enhancements to be included with my version.

Getting Started

To utilize this plugin you will need to get the plugin setup on any server you expect to utilize the plugin, so if you are in a server group, you'll need to setup the plugin on each server

The setup of this plugin requires two steps, with a 3rd optional step

1. Import the configuration forms
    a. Import the def provided in the 'Remedy Code' folder which includes a few forms and some simple workflow to ease use of the forms
2. Install and configure the plugin itself
    a. Copy the files in the 'lib' folder to your 'pluginsvr' folder on your server, or truly, any folder on your server that you would like
    b. Edit your <installdir>/conf/ar.conf/ar.cfg and <installDir/pluginsvr/pluginsvr_config.xml files and **make the appropriate changes** suggested in the files in the 'templates' folder (ensure that you don't put this new plugin inside of another plugin or pluginset tag). You have the ability to either setup the plugin in the existing 'defaultPluginServer' located in the above folder, or you can stand up a new plugin server to house this plugin on it's own, I don't provide instructions in this doc on how to stand up a new plugin server, but it's not difficult
    c. Restart the Remedy server
3. (OPTIONAL) Import the sample data and sample forms to see a working example OOTB

That's it, you are now ready to move onto the configuration that will allow you to consume Restful services through vendor forms.

Configuring the Plugin

Below settings can be configured in either pluginsvr_config.xml, or in the Config form (default name APL:RestfulVendorForms:Config). The config form always takes precedence over the xml file. The only reason you would need to configure anything from this table would if you need to add an optional configuration or if you wanted to change a default. If in a server group, items specified in the config form are automatically applied to all servers in the server group, modifications to the xml will require a server restart

Setting	Default	Details
AlwaysLogDetails	FALSE	If set to true, every call to the plugin is logged with debug level details
ArrayDelimiter	,	, is a good delimiter, unless the data you want to parse contains one, you can set this to any value you deem is appropriate
CacheForm	APL:RestfulVendorForms:Cache	If the default name does not exist, the plugin will attempt to find it based on the field ID's on the form
ConfigForm	APL:RestfulVendorForms:Config	If the default name does not exist, the plugin will attempt to find it based on the field ID's on the form
FieldForm	APL:RestfulVendorForms:Fields	If the default name does not exist, the plugin will attempt to find it based on the field ID's on the form
FunctionForm	APL:RestfulVendorForms:Functions	If the default name does not exist, the plugin will attempt to find it based on the field ID's on the form
KeepCacheSeconds	86400 (Seconds)	This is the overall default, each individual table can have it set independently, or if using the BypassCache feature, you can retrieve the results again
KeepResultSeconds	300 (Seconds)	There are two forms of cache, that retrieved from the remote system, and that which is displayed as a resultset to the user, resultsets can differ from main cache and should have a shorter life

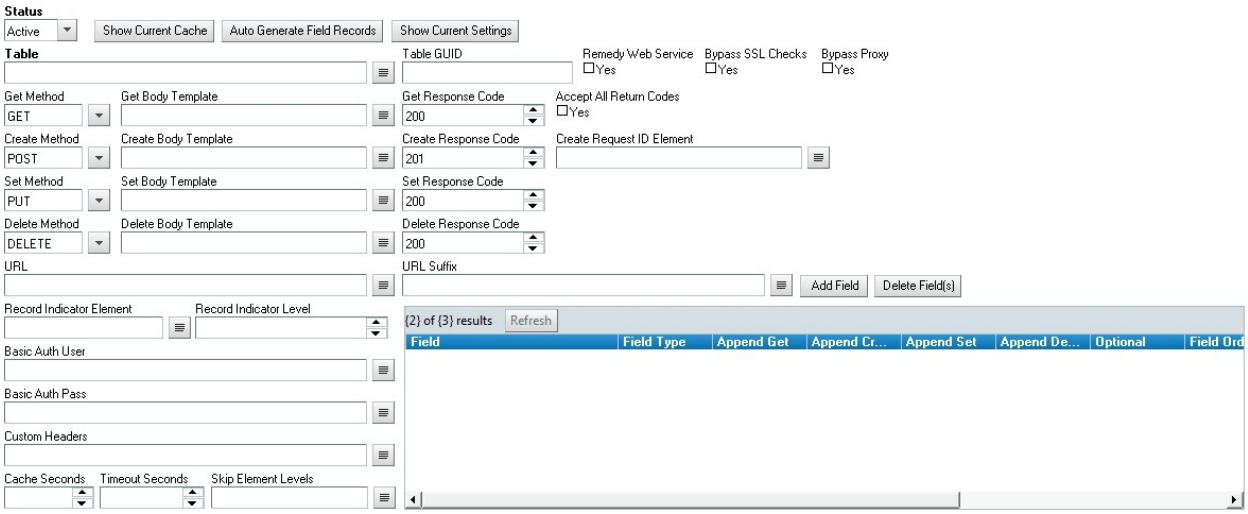
KeyStoreAlgorithm	PKIX	Algorithm is entirely dependent on the keystore in use, please set to the proper value for your keystore
KeyStoreFile	N/A	This is the location of your keystore file
KeyStoreFormat	PKCS12	Possible values could be, but are not limited to JKS, JCEKS, PKCS12, PKCS11, BKS
KeyStorePass	N/A	This is the password to get into your keystore
KeyStoreProtocol	TLS	Valid values for this would be TLS or SSL
OutputRaw	false	If set to true, the Raw output of the rest call will be provided as part of the output either to screen or log depending on situation
ProxyPass	N/A	This would be the password for the ProxyUser
ProxyPort	N/A	This would be the port the plugin will use to connect to your ProxyServer
ProxyServer	N/A	This would the name or IP of the proxy server you need to use to connect externally
ProxyUser	N/A	The user name used to connect to ProxyServer
RemedyServerName	localhost, Server-Connect-Name, Server-Name	The value starts out as localhost, but the plugin checks both Server-Connect-Name and Server-Name (in that order) and uses those if found
RemedyServerPort	0	The plugin will check server config and change this to the value specified in TCD-Port if present
TableForm	APL:RestfulVendorForms:Table	If the default name does not exist, the plugin will attempt to find it based on the field ID's on the form

TemplateFieldDelimiter	|	When building body templates you can do dynamic field replacement with field element names being replaced, you surround the field name with the delimiter, if you need a delimiter other than |, you can change it to your desired value
Timeout	90 (Seconds)	This is the overall timeout, each Table entry can define its own timeout

Configuring New Restful Service

Table Record

Once the plugin is installed and configured you need to identify what sort of web service you need, and information needed to get information out of it. This is done in APL:RestfulVendorForms:Table and APL:RestfulVendorForms:Fields (or alternatives if specified in config file.



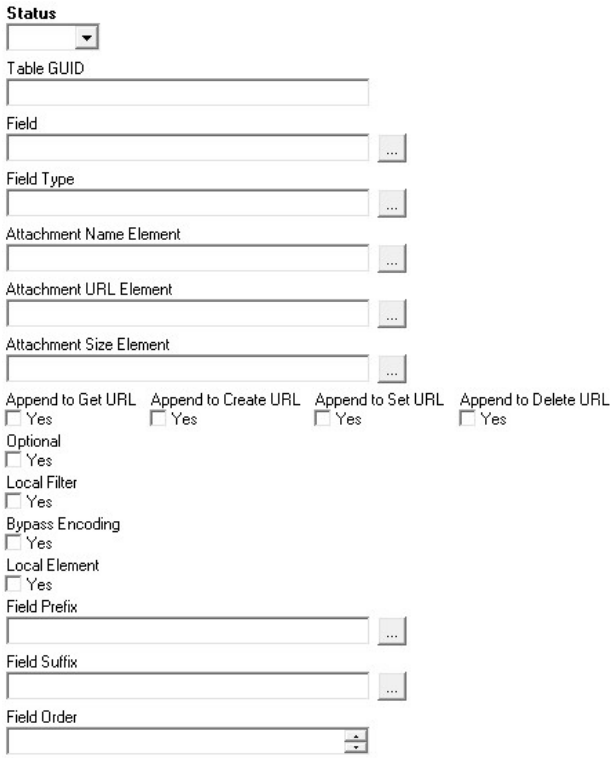
- Show Current Cache: This button will show you the current contents of the cache
- Auto Generate Field Records: After configuring the Table record, pressing this button will ask you for Sample information to make an actual call to the web service in question. After providing details (often copied out of a tool like Postman) the plugin generates a complete element list for the service and populates the 'Fields' table with that list. If this function is used on an existing service, it will only populate new field elements that don't already exist in the table
- Table: This is an arbitrary name, it is the name of the 'source' that will be used when building a Vendor form using Dev Studio
- Table GUID: This can be left blank during submit as it's assigned by the system

- Remedy Web Service: This is a special check box for use when the remote system is a Remedy 9.x Restful web services. If you check this box, the plugin will utilize the User/Password provided below to get a token that is then used to perform the query against the Remedy system
- Bypass SSL Checks: This will instruct the plugin to not do ANY Certificate checking of any sort, allowing you to use self signed certificates, if desired
- Bypass Proxy: If the plugin is configured, this option allows you to bypass use of that proxy service for the selected Table record and any vendor forms built on it
- Allow All Response Codes: This allows the plugin to handle any/all response codes that come back from the remote system as 'success', this is necessary for services that can return multiple codes that you need to be able to handle as successful
- Get: These fields will be used when you issue a query to your vendor form, this is the only way to get a result set back, even if you are doing a create on the other end, if you want to get results back from the remote system, you'll want to a query against the Vendor form
    - Method: This will typically be a GET method, but it entirely depends on the web service that you are consuming, and what they define as their method to get data
    - Template: If your Get call requires a body to be sent, this is where you would provide that information, reading above for how to put variables into the template
    - Response Code: This defaults to 200, which is an 'OK' response code. If you need to specify another code for 'success', specify it here
- Create: These fields will be used when you issue a submit to your vendor form
    - Method: This will typically be a POST method, typically requiring a template to be specified
    - Template: This is the body of the message that will be sent when creating records in your vendor form. Your template will need to provide all information the service in question needs
    - Response Code: This defaults to 201, which is a 'CREATED' response code. If you need to specify another code for 'success', specify it here
    - Request ID Element: When performing a Create operation, the service may return an ID that you want to keep track of (accessible through the $LASTID$ keyword). If this is the case, you can specify the element name that the ID is located in (same rules apply to format as a GET operation) and that ID will be returned. WARNING: Remedy requires Request ID's to be no more than 15 characters, If the system returning the ID returns a value with length > 15, unexpected results may occur
- Set: These fields will be used when you try to update a record on a vendor form. The tricky piece to this is that in order to do a Set, you must also do a Get, no matter if you are doing a push field modify, or directly on the Vendor form itself, to do a set, you must first query (push qualification), which will issue the Get methods, then once the get is done, it will then issue the set methods
    - Method: This will typically be either a PUT or a POST, but again, your documentation should provide the proper method for you
    - Template: This will be the body sent to the modify service
    - Response Code: This defaults to 200, which is an 'OK' response code. If you need to specify another code for 'success', specify it here
- Delete: These fields will be used when you try to delete a record on a vendor form. The tricky piece to this is that in order to do a Delete, you must also do a Get, no matter the method you are using to determine which record is being deleted, you must first query the Vendor form to determine the record to delete, which will issue the Get methods, then once the get is done, it will then issue the delete methods
    - Method: Typically DELETE, but your service documentation will provide you with proper values
    - Template: If needed, this will be the body of the call.

    - Response Code: This defaults to 200, which is an 'OK' response code. If you need to specify another code for 'success', specify it here
- URL: This is the main start of the URL all the way up to, but not including any of the fields that are used in the call. If https is specified normal validation of the certificate occurs, if there are any problems with the certificate, there is a check box 'Bypass SSL Checks' that will not do any validation
- URL Suffix: Anything provided in the URL Suffix will be applied to the end of the URL after any URL field/value substitutions are populated
- Delete Fields: You can select one or more field records to delete from the table as needed
- Record Indicator Element: If the web service returns multiple records, then the plugin needs to know the name of the element that defines the records. If using JSON, and the root is the loop element, then the Record indicator should be set to 'ROOT'. In all cases, the Record Indicator needs to be fully qualified, just like the field names. For an explanation of how to determine the record indicator, please see Appendix A. **NOTE: you should not specify both an Indicator element AND an Indicator Level**
- Record Indicator Level: If instead of specifying an element name as your record indicator you want to specify a 'level' of the return that is the indicator, you can specify it here, it's a 0 based numbering, for additional information, please see Appendix A. **NOTE: you should not specify both an Indicator element AND an Indicator Level**
- Basic Auth User: If your service requires authentication, this is the user that is provided
- Basic Auth Pass: If your service requires authentication, this is the password that is provided. When you provide this password, you provide it in plain text, but as soon as it is saved, it is encrypted and not visible in plain text anymore. The Plugin then uses the decrypted password when making the call
- Custom Headers: If not provided (left blank) the only header provided is an 'Accept: */*' header. If however your service needs specific headers with specific values provided, you can provide them here. This is in the format of 'Header: Value'. In the case of Service Now, it requires an 'Accept: application/json'. If multiple headers are needed, provide them one per line in this field. This field supports field replacement, read the Templating section for discussion about field replacement
- Cache Seconds: There is an overall CacheSeconds that is defined for the plugin, this is an optional per Table capability. Defining this as 0 will disable caching for this Table
- Timeout Seconds: There is an overall Timeout that is defined for the plugin, this is an optional per Table capability.
- Skip Element Levels: There are times when a Restful Service will return data that makes a standard element name not work across multiple calls, so if you need the plugin to 'ignore' certain levels of the response XML/JSON, you can specify here a comma separated list of levels to be ignored

Field Records

Once the Table record has been created, you can auto generate the fields from sample data or hit the 'Add Field' button (from the saved record, not the 'new record' screen). This will open the fields form



- Table GUID: This needs to match the GUID of the table that you are adding fields for
- Field: The easiest way to generate a field list is using the 'Auto Generate Field Records' on the table field, but if you need to generate fields that aren't part of the response URL, or for any other reason, you can create them manually. There are 5 different ways to generate the field.
    - The first way to specify a field is through the data from the source. This is the unique name of the field as it is layered in the response. In the example above 'ROOT:city:name', this response comes from JSON, so it is an element under the root, then another section named city, and then the eventual field name of 'name', that would look something like this

      {

      city {

      name: Colorado Springs

    - Instead of Response Body, you can also use Response Headers they are prefixed by Header:, so if the header you want is 'Content-Type', the field definition would be Header:Content-Type

- The third scenario is a JSON array that contains unnamed values

```
{
  "albums":[
    {
      "album":"\"Weird Al\" Yankovic",
      "songs":[
        "Ricky",
        "Gotta Boogie",
        "I Love Rocky Road",
        "Buckingham Blues",
        "Happy Birthday",
        "Stop Draggin' My Car Around",
        "My Bologna",
        "The Check's In The Mail",
        "Another One Rides The Bus",
        "I'll Be Mellow When I'm Dead",
        "Such A Groovy Guy",
        "Mr. Frump In The Iron Lung"
      ]
    }
  ]
}
```

In this scenario, the parser will do 2 different things. It will create a delimited list of

ROOT:albums:songs with value of “Ricky”, “Gotta Boogie”, etc

Additionally, to try and help you out a bit, I provide the following elements as well

ROOT:albums:songs:0 value of Ricky

ROOT:albums:songs:1 value of Gotta Boogie

And so on through the end of the array, allowing you to create a field that always pulls element ‘n’ out of an array as necessary.

- The fourth scenario involves arrays that contain named items

```
{
    "array": [
        {
            "name": "Name1",
            "value": "Value1"
        },
        {
            "name": "Name2",
            "value": "Value2"
        }
    ]
}
```

In this scenario, if you simply mapped to ROOT:array:name and ROOT:array:value, the plugin will by default, as it's running through set them to Name1/Value1 respectively, then, when it gets to the next records in the array it appends Name2/Value2 respectively and your values would be Name1,Name2 and Value1,Value2 respectively. Because you might need a specific element out of that list, I've written a subroutine that allows it. If you specify your field name as ROOT:array:name:Name1:value, when it's looping through the array, it'll look for a name/value pair in ROOT:array that has a name of Name1, and it'll look for the corresponding 'value' in that same record in the array, and will return that 'value' in the field. So, the resulting value in the field ROOT:array:name:Name1:value would be 'Value1'

- The fifth is regarding Attachments. When not receiving attachments, but instead sending them, you need the ability to specify the values associated with the attachment dynamically. When referencing the attachment itself 'attachment', but if you need to reference either the name or the size of the attachment you would use attachment:name or attachment:size respectively, where attachment is of course the name of the field in question.

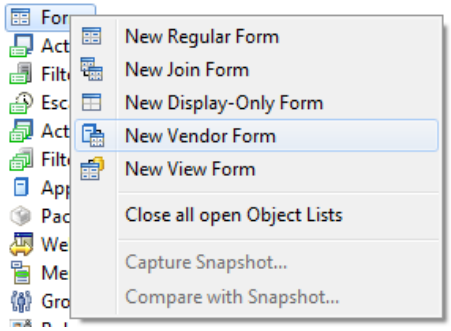
- Field Type: This will be one of CHAR, INT, DATETIME, or ATTACHMENT. If not specified it defaults to CHAR
  - Attachments are a weird piece of code to get working. There are various types of implementations on how to get a binary file from the remote system and multiple different methods to upload them as well. Depending on the implementation of how attachments are managed, I needed to build in a few different features. There are 3 Attachment related fields on the Field form
    - Name Element
    - URL Element

        - Size Element
        - Byte Element
    - Depending on the implementation of the Rest service and if you are publishing or consuming, you may have between none or all of these values populated. In the case of consuming Remedy Restful services, you may be provided up to 3 elements in the JSON response. In some of the other implementations I looked at you are only given one, the URL that the attachment is at, which includes its name. In the case of having URL only, which includes its name, you can specify the same element name for both fields and the plugin strips the name out for reference. In this case however the plugin has no way of knowing how large the file is, so the file will always be 0 bytes in the attachment field on the Vendor form, but when you get the attachment from the remote host it'll be the proper size. In the case of submitting/updating attachments on remote hosts through this plugin, the 'name' of the attachment as it's put in the remote server is derived from the Attachment Name element defined in the field record.
    - Because of how Remedy works, you can't do a 'Query' with an attachment field populated, so if you are wanting to update an attachment in a remote system, you will need to either be doing a Remedy Create or a Modify. Depending on the needs of the remote system, it may be necessary to create a Table entry and a specific Vendor form to allow for creation of records with attachments. For example, when doing create with attachments on Remedy you must do a multi-part body where the header has a content type of 'Content-Type: multipart/form-data;boundary=<boundary identifier>', and must provide a multi-part body that contains both field/value pairs, but a second section with the actual binary data, this is referenced in the BMC docs (https://docs.bmc.com/docs/ars91/entry-formname-609071438.html#post-341682506). This could in theory be done in the same Table/Vendor combination if you had Content-Type be a dynamic header that you set differently on get vs create for example, or use different Vendor forms with different header setups, either way, it's entirely possible to get it working.
    - WARNING: The plugin does Base64 encoding on the attachments as they go out, so you must specify the proper encoding in your multi-part body (Content-Transfer-Encoding: Base64). The only caveat to this is if you choose 'Bypass Encoding' on the attachment Field record, in this case the encoding of the attachment values is bypassed and sent as raw
    - Adding Vendor attachment fields turned out to be an interesting experience. I'll skip all the trials and tribulations I experienced during my efforts to try and figure out how to do. If you Vendor form has an attachment field defined, you can add it during the create process, or by a right click on the form. Either way, once it's added you won't be able to visibly see it on the form. Now that you have it on the form though, create an attachment pool through a normal right click process. Select the attachment field through the form outline view, this will make it show up on the form highlighted by nothing more than its move handles, you can then drag and drop the attachment field onto the attachment pool where it shows up like you would normally expect. I have come across a few customers running Dev Studio 7.6.04 and any time they try to save the attachment field into the pool it gives them an error and they are unable to save the form. While working with them I was able to take an export of their form without the attachment field in it (but with the attachment pool) and manually through the def, add the attachment field and set its parent field id to the attachment pool, then import that def. It's an ugly option, but it works and it's the best you can expect on that version because the vendor isn't going to be fixing that problem. If you need an example of what an attachment field looks like in def on an attachment pool, please use the sample form I have provided.
- Append to Get URL: If this field value should be used in the URL the query action, check this box
- Append to Create URL: If a field should be used in the URL of the Create action, check this box
- Append to Set URL: If a field should be used in the URL of the Set action, check this box

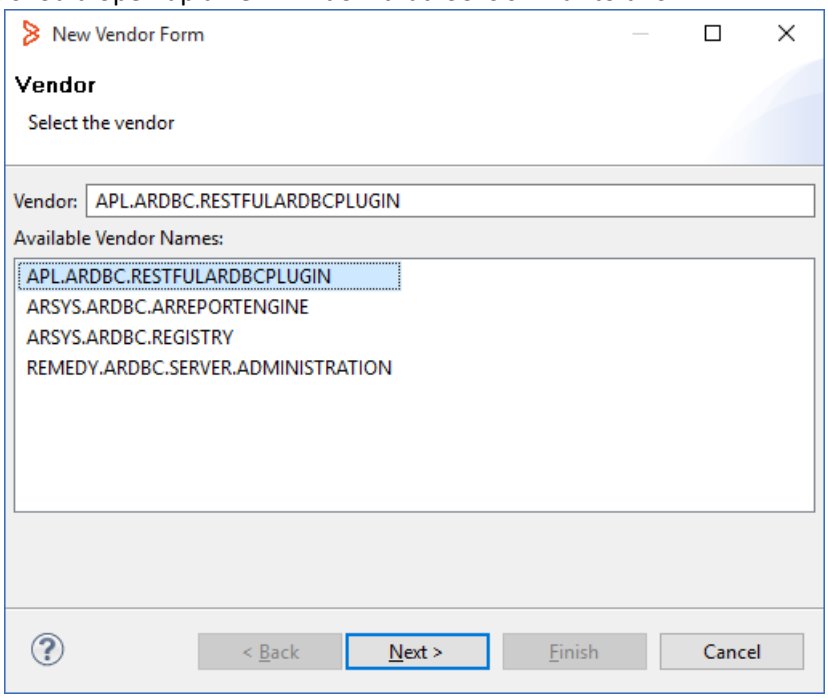
- Append to Delete URL: If a field should be used in the URL of the Delete action, check this box
- Optional: If the field is defined to be on the URL, or defined in any of the template values, but you only want it used if the value is not blank, check this box. Otherwise it'll show up.
  WARNING: If defined in a template, and that field is the last element in a list of fields, then removing the entire line will cause an improperly formed JSON (because the line above will have a ',' in it, so, ensure that this doesn't happen by forming your template so that even optional fields being removed won't cause an improper JSON object
- Local Filter: This is utilized when you don't have the ability to limit the result sets based on the service operations. For Example, you can query the groups, but it returns all groups, both active and inactive, and the service doesn't allow you to specify only the active ones. So, with this feature you can define the status field as a local filter, and when you issue the web service call, all records will still come back, but the local filter will only show the records in the resultset that you specify. I made every attempt to parse the qualification properly and only show you what you need, but there could be situations that I didn't consider, if you experience issues with values not displaying as expected, reach out with the exact scenario and I can try to identify the cause of the problem.
- Bypass Encoding: If you need to build your own JSON/XML inside Remedy and provide it in the template, then the plugin automatically encoding your stuff won't work for you, this checkbox allows you to bypass the plugin encoding everything inside the value to be 'safe' and allows you to specify your own values, alternatively, if you use this value on an attachment field, the attachment won't automatically be Base64 encoded for transmission
- Local Element: By default, an element will only be in the results if it comes from the remote system as a response to the restful call, if you however want to specify an element as part of the transaction, and include its value in the results as well, you would check this box and anything you put in it will show up properly in the results
- Field Prefix: If this field is being appended to the URL, then this is what goes into the URL before the value provided in the query
- Field Suffix: If this field is being appended to the URL and something needs to be put at the end of the field value (such as a closing > in the case of an XML input, this is what would go after it
- Field Order: Sometimes the order of elements in the URL is important, other times, it's not. This field allows you to specify which order a field will show up in the URL. One example would be a required 1st element might need to provide ?field=, whereas all of the other would need to provide &field2=…so, the first one needs to go first to ensure that the ? is in the proper place in the URL

Vendor Form Creation

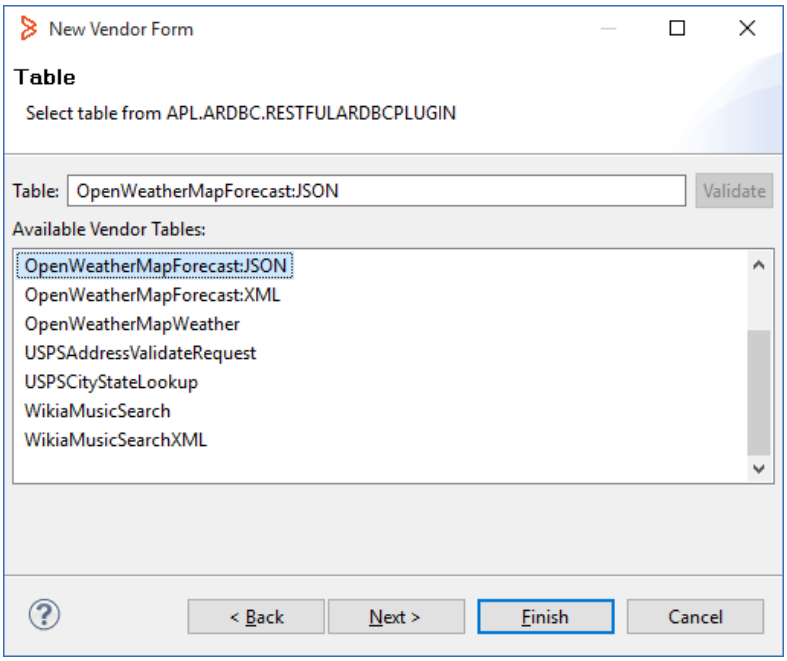
At this point you have a valid Table record, and the associated Field records created. It is now time to build the Vendor form that utilizes the records you just defined.



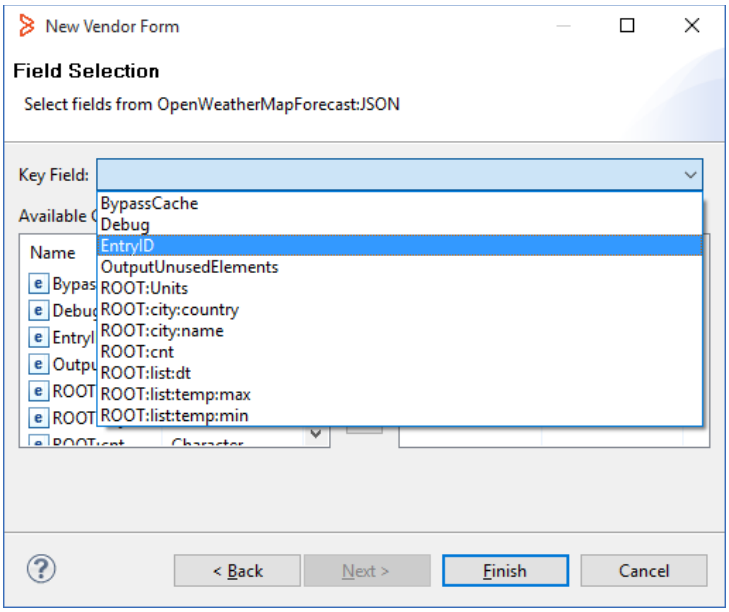
This should open up a new window that looks similar to this



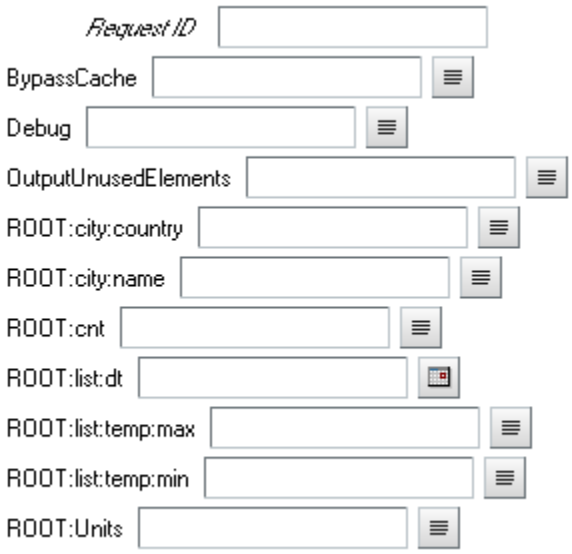
You will choose the APL plugin. This will provide you with a list of Tables that you have defined in the system



You can choose the table being built



When building your form, you must use the built in field 'EntryID' as the Key Field. Failure to select that as your Key Field will result in the plugin not functioning properly. You will then add in any/all of the fields you want to use.



Once built, you should have a form like this. You can rename these fields as you see fit and build out the display in any way you see fit. Your Vendor form should now be ready.

Consumption of Remote Services

People usually equate performing a Query in Remedy, to performing a GET in the remote system, and Performing a Create in Remedy to doing a POST in the remote system, but the two are not

necessarily related to each other. There are times when they match up and you can do that without an issue, but there are also times where they don't match up, I'll discuss those times here.

Performing a Query in Remedy simply means you are doing a GET in remedy, this can be through either a Table Field doing lookup on your Vendor Form, or a piece of workflow that does a set fields from it. Maybe even an escalation that runs against it. When doing a query in Remedy you have the advantage of being able to get results from the remote system. So, if you need to be able to see what the output of the remote restful call is, you'll want to typically do a query, even if the remote call is a POST or something similar. The only limit that exists in Remedy when doing queries is that you can't provide an attachment field as one of the query fields….so, if your intent is to transfer an attachment to the remote system, you will need to do a create/submit.

Performing a Create in Remedy has the distinct advantage of the ability to provide attachments as part of your payload, but it doesn't allow you to get more than a single element back. This goes way back into the history of Remedy, but when you Create a record in a form, you have the ability to receive the 'Request ID' of the record you just created. Because of this limitation, if you are needing to be able to get things back, I've provided the ability to get the value from a single element back from the remote system.

Set and Delete also work within the plugin, but are significantly more difficult to work with because within the Remedy framework, to do a Set (Push workflow with update option) it must first do a Query (using the Qualification fields) to the Vendor form, return valid results, then perform a second restful call that then uses the push fields as the fields for the second call, generally speaking, it's just easier to setup multiple vendor forms and make multiple calls than trying to get Set and Delete to work.

Assuming you are doing a Query in Remedy, regardless of the method being used for the Rest services, it's assuming you want to get some results back. This plugin is configured to handle 4 different scenarios of response:

- XML: The plugin assumes that any response that starts with a < is an XML. Field 'names' are a combination of all of the element names, so if you had an xml structure like this

  ```
  <Company>
   <Employee>
     <FirstName>Frank</FirstName>
   </Employee>
  </Company>
  ```

  Your field name would be Company:Employee:FirstName with a value of Frank

  - NOTE: You may sometimes get a SAXParser error, because the response coming back is actually HTML instead of XML. This typically only occurs when the remote system encounters an error condition, and instead of sending a machine readable XML response, it sends an HTML formatted response intended to be read in a browser. If you encounter this

scenario, you should work with your restful provided and have them provide either an XML or JSON responses.

- JSON: The plugin will accept a string that starts with either { or [as JSON. { being the start of a JSON document, and [being the start of a JSON Array. Being that JSON doesn't have a root element name, the plugin calls the root element ROOT, so in the same document structure we had for the above XML example, but this time in JSON

```
{
  "Company": {
    "Employee": {
      "FirstName": "Frank"
    }
  }
}
```

  We would have a field named ROOT:Company:Employee:FirstName with a value of Frank

- There are also times that you will make a call and the return isn't record data, but instead it's just a single piece of information. This is true either of the below situations:
  - Making a call to get a Token: In this scenario, there are some services (Remedy Included) that when you make an appropriate POST call to the correct URL, the response is just a token, it's not inside a JSON doc with field/value pairs, it's just the text. In this scenario you would want to create a single field record and name it ROOT. When you put that field on your Vendor form, if the resulting output has no document, then it converts the value to a String object and returns it as the only field returned
  - Downloading an Attachment: If the URL you are accessing is instead a file, then the return isn't a string at all, but instead it's a binary result. To handle this scenario you will need to do a bit more work. You'll need to create a field named ROOT, and check the 'bypass encoding' check box, this will tell the plugin not to convert the bytes into a string. Then you will need to create another Field definition of type 'ATTACHMENT'. Inside this definition you'll want to populate the Attachment Byte Element with the word ROOT. This tells the plugin to put the attachment bytes into the attachment field. You can also create another field that you could put into the Attachment Name Element so that you have the ability to name the file you are downloading

Template Field Name Replacement

With the use of templates, there needs to be the ability to essentially do find/replace at run-time. Templating works in both the Body section as well as the Custom Headers. By default, the delimiter is |, but can be changed as needed by setting the TemplateFieldDelimiter config parameter. If you wanted to send the following to remote system

```
{
    "field": "Value",
    "field2": false
}
```

We need to translate that into something that the plugin will replace with run time values. So, let's assume you have a field on your form that is mapped to element ROOT:Field1, and another mapped to ROOT:Field2. So, your template would look like this

```
{
    "field": "|ROOT:Field1|",
    "field2": |ROOT:Field2|
}
```

Additionally, if a field is marked as 'Optional' in the field definition, and that field is defined in the template, it will remove the entire row that contains that field, so in the case above, if ROOT:Field2 were marked as optional, and not provided, it would be removed, which would actually make the JSON invalid because the last element would have a , at the end, so you are responsible for ensuring that your JSON is valid after the templating process is done.

Token Authentication

One of the most common types of rest authentication is via a token of some sort. This token is typically retrieved by providing a username/password at a given URL and the output of that call is then used as the input of a secondary call and is typically part of the header of the subsequent calls. This type of process is done automatically when you select the 'Remedy Web Service' checkbox, but if your remote system needs to use tokens and isn't a Remedy service, you can still do this work. You will likely need to setup two table entries, one with the proper URL to get the token, and a second with the proper URL and setup to utilize the token. You may want to setup additional forms and workflow to manage the Token and retrieve new ones when they expire. If the remote system generates a token response that is not in JSON/XML format, Read "Making a call to get a Token" above

Debugging and Tweaking

If you are having trouble getting the plugin up and running, please ensure that all modifications were made to the ar.conf/ar.cfg and pluginsvr_config.xml files with the appropriate replacements of template values with actual server names and locations of the files in question and ensure that the jar files made it to your server unmolested.

During initial configuration of your form, you may find that you make changes to a form definition, but the results are not being reflected in vendor form. This could be due to the caching mechanism in use. Any time you add fields or change the record indicator, the results cache will need to be rebuilt. This can be done by populating the 'BypassCache' field with a value (anything other than blank) and the

request will rebuild the result set and re-populate the cache, or you can remove the records from the Cache form manually.

There are several default fields that can exist in any Vendor form built from this plugin and each has its own function. Each of the special fields needs to exist on the form only if needed, and to activate the function of the field, needs to not be null. i.e., if you have the BypassCache field on the form, but don't provide a value, the cache is used, if you put something, anything in that field, the cache is bypassed.

- BypassCache: If used, the query will not utilize cache and will instead make a call to the external system to get the data. Cache is only used if the same query was executed prior, and the query hasn't expired out of the cache.
- OutputUnusedElements: This is a debugging effort to allow in building out the form once it exists. Sometimes it's possible that the external system has added/modified elements of the response, and the easiest way to determine what they are is by analyzing the output. If this field exists and is populated, the output for the plugin server will contain any elements existing in the output that are not mapped to fields in the Vendor form created.
- Debug: This will cause additional detailed information to populate into the arjavaplugin.log file for the server running the plugin. This allows more detailed understanding of how the query is being performed as well as what's happening with cache, count of results and such. All of the non-error based logging performed by the Debug option is done at the 'INFO' level. If you want to be able to see this detail, you will want to change your logging levels to INFO from the default of 'WARN'. If you have the 'AlwaysLogDetails' config parameter configured, this option is not needed.
- ErrorOutput: If present, this field will be populated with anything that is considered an 'error' by the plugin. These are typically also output into the arjavaplugin.log file, but if this field is present, also put into the results of your query, thus allowing you to control the workflow results based on plugin errors. If this field doesn't exist any errors returned by the plugin are thrown as Remedy errors and can be handled by error handlers as normally done through Remedy
- ReturnCode: If you choose to add this field to your Vendor form, you will get the return code of your Restful call populated into this field, this is useful for troubleshooting wayward services and you need to be able to handle all the possible codes individually

Release Notes

4.13.3 – Sep 29, 2023

- Upgrade Requirements: This release is only updating the code in the plugin itself, so only the Jar is needing to be replaced
- BUG: Fixed issues with arrays not populating missing elements
- BUG: Fixed issue with Local Filtering that wasn't being cleaned up between calls, causing local filtering from one call to affect another call on the same thread
- BUG: Fixed NPE encountered when the table cache seconds is null instead of ANY number
- ENHANCEMENT: Added additional logging around Local Filtering to help identify issues, when they come up
- ENHANCEMENT: Added 'OutputRaw' configuration parameter to help troubleshoot some troublesoom rest services that aren't returning the expected output
- MAINTENANCE: Updated JSON jar file to json-20230618, latest as of the moment of writing

4.13.2 – Sep 18, 2021

- Upgrade Requirements: This release is only updating the code in the plugin itself, so only the Jar is needing to be replaced
- BUG: When providing a result to Remedy, if the remote system didn't return a value I didn't create a value for that field in the Entry. This should have resulted in a null value in that field because no value was being provided, but apparently the Remedy server was using other values instead of the not null value provided, and causing issues in some situations. I have modified the code so that if no value is provided by the restful service, but the field exists on the Vendor form, that I provide a null value object, this seems to have cleared up the issue.
- BUG: If the field was defined as an int, but the remote system provided a null, and the field was used as a local filter, then it was throwing a null pointer exception.
- BUG: Altered how the JSON Processor was handling 'null' values. It was treating them as strings with a value of null, now, it doesn't process the value, which should cause it to come up as a Remedy null object because no value was returned in the json
- BUG: Modified Local Filtering a bit to ensure that proper filtering is happening when non string values are used
- BUG: Fixed JSON parser issue where if the root was an array and the array was unnamed elements, it wasn't properly parsing the results as rows
- ENHANCEMENT: Added the processing of Cookies as field elements. This allows for using Cookies as authentication. The new fields are populated as Cookie-<cookie name> allowing you to use those values.

4.13 – Mar 6, 2021

- Upgrade Requirements: This release updates both the JAR and the form definitions, so you'll want to re-import the def file.

- ENHANCEMENT: Added the ability to pull straight text from the remote system. Previously the return response needed to be either JSON or XML, but now it can accept just plain text. This is useful when using the plugin to get tokens from remote systems that don't return them in a JSON document.

- ENHANCEMENT: You can now download attachments directly with the plugin. This one goes along with the ability to do things that aren't JSON or XML. Previous iterations of the plugin required you to get a response that gave you details about the attachment itself like where the file was, it's name, it's size, that sort of thing, but now, if the URL you have is the attachment itself, you now have the ability to get that attachment with this plugin

4.12 – Nov 21, 2020

- Upgrade Requirements: This release updates both the JAR and the form definitions, so you'll want to re-import the def file.
- ENHANCEMENT: Updated attachment code so that if the attachment URL is relative instead of fully qualified, it auto appends the service URL root to the beginning of the URL to retrieve it.
- CHANGE: I updated the json library that was in use with the tool, not because I experienced any issues with the old one, but because it was now about 3 years old so I figured I would grab a newer one, if you don't choose to update your copy with this one, that's ok
- BUG: Fixed a NullPointer Exception in the getCharacterSet method
- BUG: When Remedy was providing the body template, it was only providing a line feed between lines. Had a customer who's remote system is expecting a full carriage return/line feed, which it turns out is what the RFC specifies...so, I modified it to ensure that the body contains full CRLF instead of just LF
- BUG: In version 4.2 I enhanced the bypass code to include Attachments, but apparently didn't get it quite right. So I now have it right and it's sending over the attachment bytes 'raw' with a Content-Type of BINARY

4.11 – Feb 12, 2020

- Upgrade Requirements: This release is only updating the code in the plugin itself, so only the Jar is needing to be replaced
- BUG: Changes made in 4.7 made the plugin stop working on 8.x machines, causing an 'Incompatible data types for intended relational operation' error to be thrown. I have fixed that error
- ENHANCEMENT: One customer was experiencing NullPointerException problems, while I wasn't able to fully identify what was causing the NPE, I have modified the code to ensure that the NPE won't cause errors in the plugin any more

4.10 – Dec 15, 2019

- Upgrade Requirements: This release is only updating the code in the plugin itself, so only the Jar is needing to be replaced
- BUG: Found a bug in the cache management that was allowing cached results that should have been expired to be deleted, causing them to be re-used after their cache expiration date, I have corrected
- ENHANCEMENT: Previously if the Rest call ended up in an error on the remote end I would just throw the error to you, now I'm catching the error and getting the error output, and if you have

your system defined to accept all return codes, you can then get access to the error text and utilize it

4.8 – Sept 29, 2019

- Upgrade Requirements: This release updates both the JAR and the form definitions, so you'll want to re-import the def file.
- BUG: There was a bug in the optional field template regex that was replacing the body with blank instead of just removing the optional null element
- BUG: The plugin uses 'KeepResultSeconds' as the value, but the config file was improperly calling it 'KeepResultsSeconds', so values used in the xml weren't being properly recognized. I've verified at this point that the docs and all references to this are singular
- BUG: Due to a change between 4.6 and 4.7, and a problem with reading the plugin config, you weren't able to change any of the default behaviors. So I disabled the offending workflow and updated the code to properly read the config values from the XML

4.7 – July 12, 2019

- Upgrade Requirements: This release updates both the JAR and the form definitions, so you'll want to re-import the def file.
- BUG: Fixed some display issues when using the Auto Generate Fields function
- BUG: In 4.5 I introduced the Config form, but the ability to override the 'form' attributes wasn't working fully
- BUG: KeepResultSeconds wasn't working properly from the config form
- BUG: In 3.8 I added padding to make the request id 15 characters to fix a join compatibility problem. In 4.5 I changed the format of the requestid, initially it was a 10 digit timestamp, and then the number of the returned record, I then changed it to include the transaction id, this unfortunately caused it to go over the 15 limit, which caused additional issues, so in this release I'm removing the 10 digit timestamp, keeping the transaction id, and keeping the padding, so the format now will be 123400000000001, where the transaction id and the record of the request. This should hopefully fix the various problems that people are having
- ENHANCEMENT: In prior versions of the plugin, the plugin was doing quite a few calls back into the Remedy server every time you performed any transaction, more than was truly needed, I refactored some of the code to do the work local to the plugin instead of asking the AR Server to do the work. This has a twofold benefit, first it makes the plugin faster, second it reduces the load that the plugin places on the AR Server
- ENHANCEMENT: Refactored much of the config retrieval code to ensure that all settings are properly logged and used in the proper order (xml, then config)
- ENHANCEMENT: Added the ability to get the current running parameters from the Table form

4.6 – July 7, 2019

- Upgrade Requirements: This release updates both the JAR and the form definitions, so you'll want to re-import the def file.
- BUG: The response code wasn't showing up properly, corrected and verified
- ENHANCEMENT: There are times that you want the value that you query to show up in the results, even if it's not coming from the remote system. I decided to call this a 'Local Element', if you have this situation, you can create your element on the Fields form and mark it as a Local Element, and the value provided in the query will show up in the results set just like everything coming back from the remote system

4.5 – May 19, 2019

- Upgrade Requirements: This release updates both the JAR and the form definitions, so you'll want to re-import the def file.
- BUG: Fixed a NullPointerException in the 'SetEntry' portion of the code
- BUG: The cache was not properly identifying unique requests causing possible cross call contamination. This change required the change to the Request ID format and modification of what's stored in the 'URL' field in the cache form.
- ENHANCEMENT: I have added a Config form that allows you to configure any/all configurable items via record definition instead of reconfiguring the pluginsvr_config.xml. The plugin still reads from the xml file, but everything in the config form overrides anything in the xml, additionally, because the config form is accessible to all servers in the server group, you can configure in one place and it immediately takes effect on all servers without the need to restart, which is a huge bonus

4.2 – January 30, 2019

- Upgrade Requirements: This release updates both the JAR and the form definitions, so you'll want to re-import the def file.
- BUG: When validating JSON document, if the document starts with a [instead of a {, the plugin throws an error and prevents the call from completing properly
- BUG: When a field was in a template and wasn't populated, despite not being marked optional, the plugin was removing the field from the template, not it just removes the element name leaving it as a blank value
- ENHANCEMENT: Extended the 'Bypass Encoding' function to the attachment field. Prior to this release the attachment field was always base64 encoded, had someone ask for 'raw' attachment data, so I made the encoding optional with a checkbox on this field type
- ENHANCEMENT: Added an 'Accept All Return Codes' checkbox for services you may encounter that you always want to consider the return a successful
- ENHANCEMENT: Added 'Return Code' as a field available on Vendor forms so that you can interpret that at your will
- ENHANCEMENT: Added ability to specify array parsing delimiter. In prior releases, an array was assembled using , as a delimiter, as this can be an actual value in the array, a request was made to make the delimiter configurable, you now have that ability
- CHANGE: Modified the 'Table' field on the Table form to be required, some people were apparently leaving this field blank and causing themselves issues

4.1 – October 8, 2018

- Upgrade Requirements: This release updates both the JAR and the form definitions, so you'll want to re-import the def file.
- BUG: When I wrote the Array Parsing portions of the code I only wrote them to work with JSON, I had a user need the same functionality for XML results so I refactored to include XML in that effort
- BUG: When writing the code for the keystore, I wasn't properly pulling the KeyStoreProtocol from the config file, this has been corrected
- ENHANCEMENT: When I added automatic encoding of values I removed flexibility in allowing you to do everything yourself, so in this release I'm adding a 'Bypass Encoding' checkbox to fields so that the values you put in that element won't be automatically encoded, so you can build your own JSON/XML without the format getting screwed up

- ENHANCEMENT: Had a user contact me regarding the need to send a 'PATCH' instead of GET/POST/etc, apparently Java complains about certain methods that aren't 'standard', so I needed to add code in allowing it to go through
- ENHANCEMENT: Enhanced the query processor to handle Boolean queries (1=1, 1=2, etc) and return no entries on a Boolean false instead of issuing the Query Restful service
- ENHANCEMENT: Made the value that is used as a delimiter a configurable parameter, so while the default remains | you can now specify your own value in your templates per the plugin config example

4.0 – August 11, 2018

- Upgrade Requirements: This release updates both the JAR and the form definitions, so you'll want to re-import the def file ensuring to remove fields that don't exist in the def as I have removed at least one field. Additionally I've changed JSON libraries so you will need to ensure that your pluginsvr_config.xml is updated properly with the new library being put in place on the server as well
- CHANGE: A major change in this release was a complete rewrite of the JSON Parser using a new library. This rewrite allows for greater flexibility in handling of the JSON and increased capabilities, additionally, the size of the JSON library I'm using is significantly smaller than the original one, this requires reconfiguration of the plugin to utilize the new library, all of the sample files have been updated appropriately
- ENHANCEMENT: I've modified the tool to automatically detect XML/JSON responses, so no longer have a field requesting this information from you, that's one less thing you need to configure for your services
- ENHANCEMENT: Added the ability to specify a record indicator level instead of an element name, this provides greater flexibility in defining your records
- ENHANCEMENT: Added ability to 'ignore' certain levels of the returned results. See Appendix B for more details

3.10 – July 29, 2018

- Upgrade Requirements: This release updates both the JAR and the form definitions, so you'll want to re-import the def file
- CHANGE: In an effort to make the setup and configuration easier and to utilize information already available to the plugin, I've removed a few settings: BackupUserName, BackupUserPass, BackupRASHash, and RemedyServerPort. The 'Backup' values are unneeded because of access that the plugin already has to the RASHash in the ar.cfg file, asking you to provide it for the plugin was not necessary and I've coded the plugin to just simply use that information automatically instead of forcing you to provide it, similarly, the port that Remedy is running on is also available and un-needed. Despite having access to the ar.cfg settings I've decided to leave the RemedyServerName setting because it might be necessary for flexibility
- ENHANCEMENT: In version 2.7 I added the ability for the plugin to use a Proxy server, but with that setting I made it an all or nothing setting, either the plugin was using it or not, not accounting for the fact that the plugin might need to use it for some but not all services being called, this update adds a new checkbox on the Table record allowing you to 'opt out' of using proxy on select services if a proxy server is configured
- ENHANCEMENT: I had a user contact me regarding the ability to get a value out of a response header instead of the body, it takes a slightly different nomenclature in the format of the field, but it works

3.9 – July 13, 2018

- Upgrade Requirements: This release updates both the JAR and the form definitions, so you'll want to re-import the def file
- BUG: The enhancement I added in 3.7 didn't work properly, I've done a bit more rigorous testing of it and it seems to be working better, thank you for the report
- ENHANCEMENT: I've modified the 'Cache' form to no longer store Status History. I did some analysis of the performance of the caching function and the status history function was not needed and caused unnecessary DB interaction, disabling this feature should enhance the performance of the plugin by a non-insubstantial amount

3.8 – June 22, 2018

- Upgrade Requirements: This release is only updating the code in the plugin itself, so only the Jar is needing to be replaced
- **CHANGE**: Previously, the EntryID of any given cache record was the epoch timestamp + the request id. This works fine in most circumstances, It was however recently discovered that when using Vendor forms in Joins, that the vendor form must produce a 15 character entry id for all records. This is a minor change in the code, but important for functioning of the vendor form in question within Joins, so I needed to make this change for a specific user.
- **CHANGE**: While doing some checking today I found that I wasn't using the Jackson Annotations library anymore, so I've removed it from the download and removed it from the templates. This will NOT impact your system if you continue to have it on your server but it's no longer necessary

3.7 – June 20, 2018

- Upgrade Requirements: This release is only updating the code in the plugin itself, so only the Jar is needing to be replaced
- ENHANCEMENT: Prior to this release, it was incumbent upon you to ensure the values that you were sending out were compliant with the formatting requirements of the format you were using, so if you had quotes or < characters or something along those lines in your data, the plugin did nothing to help you out, I have enhanced the template portion of the process to replace 'special' characters that would normally screw up the transaction with their relevant escaped/converted values for you
- ENHANCEMENT: Updated the JSON Parser code to attempt to detect the character encoding of the response JSON. According to specification this should always be in UTF-8 encoding, but apparently sometimes people don't follow standards and it messes with other people assumptions, this is an attempt to accommodate those systems
- **CHANGE**: Within a given record, if the record contained an array with named values in it, you had a few options, you could specify which field/value pair you wanted, or the plugin would simply give you the last one. I've had a request to provide the capability to store all of the field/value pairs for all elements in the array, so what I have done is changed the behavior from just giving you the last one, to appending each of them sequentially in the element separated by a comma, this should allow you to collect all of the data and parse it out manually once the data is retrieved

3.6 – April 7, 2018

- Upgrade Requirements: This release updates the jar and some of the Remedy code. So, replacing the jar and import the def file replacing all of the objects.

- BUG: When you are sending a Date/Time field from the vendor form outside to another system I wasn't making provision to convert that back from a Remedy value to a human readable value, I've added that provision. It uses format 2018-04-07T14:02:00
- BUG: There were times when a field wasn't being used, and it was marked optional, but the field was still showing up in the body of the message being sent over, causing issues, this has been corrected
- **CHANGE**: The former default behavior from the beginning was to bypass any/all ssl issues with the certificate on the remote end, then in 3.0 I had a handshake error when using part of it, likely related to the fact that I was not doing any ssl validation, so I commented out part of the ssl all accepting code, which caused issues for others where the ssl cert was not valid for the host in question, so I have now added a check box on the table form that allows you to bypass all SSL cert checks...it's defaulted to NOT checked, which means that all existing configurations will now validate the SSL configuration instead of the default before of NOT validating them, so if your SSL in question has 'issues', you can check the box and it won't do any validation, otherwise you will do normal, proper validation.

3.5 – March 22, 2018

- Upgrade Requirements: This release is only updating the code in the plugin itself, so only the Jar is needing to be replaced
- BUG: The web connection setup process was not setting the method properly if the remote server was a Remedy server

3.4 – February 19, 2018

- The only change in this release is the change of the license from GPL 3.0 to Apache 2.0

3.2 – December 8, 2017

- Upgrade Requirements: This release is only updating the code in the plugin itself, so only the Jar is needing to be replaced, but because of some minor changes in the supporting def, you can re-import that if you care to
- BUG: Slight issue in the template pluginsvr_config.xml, missing a / where one was needed, corrected
- BUG: When making Rest calls to other Remedy servers it was incorrectly trying to open the connection to the remote server twice
- BUG: Made a slight modification to the RemedyUser sample Table entry, you may want to remove and re-import that arx file if you are using/testing that example
- BUG: ProcessEntry function wasn't verifying that the entry in question wasn't null causing nullPointerException
- CHANGE: I added default response codes to the Table form, this doesn't change the defaults that the plugin uses, it just sets them in new records created by default

3.1 – November 5, 2017

- Upgrade Requirements: This release is only updating the code in the plugin itself, so only the Jar is needing to be replaced
- BUG: Because of some refactoring I needed to do for the Attachment work I messed up some of the code for header replacement and Remedy authentication

3.0 – October 28, 2017

- Upgrade Requirements: This release updates the jar and some of the Remedy code. So, replacing the jar and import the def file replacing all of the objects. There are also a number of

new config file parameters that can be utilized, so look at the template and see if any of them are relevant to your needs

- **CHANGE**: In version 2.4 when I upgraded Jackson from version 2.5.4 to 2.8.5, they apparently upgraded their java requirement from 6 to 7, which effectively upgraded mine as well, but I didn't change the requirement at the package level...so the plugin would still load, but when it tried to use the Jackson libraries it would fail version check...so, since 2.4 you needed Java 7, but now the plugin requires it too
- BUG: My use of the 'all trusting ssl' code was causing an SSL Handshake error. After analyzing the effort it was related to the HostnameVerifier section, commenting that out allowed the connection to succeed, and verifying a remote system's hostname isn't a bad thing in general, so I left it commented out
- BUG: Added Error checking around custom headers to prevent index out of bounds errors
- BUG: Some of the workflow associated with automatic field generation was pointing to the wrong fields, I've fixed this
- BUG: Local Filtering was allowing values that were 'null' to show up when filtering on a specific value
- ENHANCEMENT: Added support for Attachment fields, see the rest of the documentation on how to build/manage/use an attachment field
- ENHANCEMENT: Converted the connection so that it properly recognizes http vs https for connection types enabling better handling of SSL specific capabilities
- ENHANCEMENT: Added the ability to set a keystore. The purpose of this keystore is to allow the client to provide a client certificate for authentication, templates updated and documentation updated to reference new xml parameter options
- ENHANCEMENT: The 'Custom Header' is now dynamic, it follows the same rules as the template bodies, you now have the ability to specify a field value in the headers and have that value replaced by the actual value at run-time
- ENHANCEMENT: Rest cache was previously stored in memory, which meant that if you were using this in a server group, each server would have it's own cache of the values. I have converted the plugin to store the cache in a new cache form which will provide more stable caching across server group situations

2.9 – May 8, 2017

- Upgrade Requirements: This release is only updating the code in the plugin itself, so only the Jar is needing to be replaced
- BUG: Issue identified in some JSON cases one 'record' was not logged into the recordList properly

2.8.1 – May 1, 2017

- Upgrade Requirements: This release is only updating the code in the plugin itself, so only the Jar is needing to be replaced
- BUG: In version 2.5 I replaced the SSL Trust Manager with the 'Extended' trust manager in an attempt to fix something with the Remedy authentication, that ended up not being the issue, but I didn't revert the change...this change had the unintended consequence of requiring an upgrade to Java 7. I have reverted to the old trusted store code which restores Java 6 compatibility.

2.8 – April 27, 2017

- Upgrade Requirements: This release is only updating the code in the plugin itself, so only the Jar is needing to be replaced
- BUG: Corrected an 'Already Connected' error when the body was being used

2.7 – April 14, 2017

- Upgrade Requirements: This release is only updating the code in the plugin itself, so only the Jar is needing to be replaced
- BUG: If you had 'Remedy Server' checked, but your URL didn't have a port specified, the URL wasn't built properly and didn't work
- ENHANCEMENT: Had the first server that needed a proxy server, added the config and code to handle it
- ENHANCEMENT: Added config parameters allowing the Table and Field form names to be able to be specified, allowing the user to rename the APL forms on their servers as they feel appropriate without causing the plugin to fail
- ENHANCEMENT: Added config parameter AlwaysLogDetails, if set to true, all activity is logged without the need to set the debug flag on a particular call, it's still logged at 'INFO' level

2.6 – March 23, 2017

- Upgrade Requirements: This release is only updating the code in the plugin itself, so only the Jar is needing to be replaced
- BUG: Fixed problem when passing 'null' values into fields
- BUG: Corrected RequestID value when the 'create' was not successful
- ENHANCEMENT: Added code to Template processing to remove all 'non processed' template values and replace it with blank strings
- ENHANCEMENT: Modified the code to provide a 'Content-Length' header in all situations, not just when a body template is provided

2.5 – December 15, 2016

- Upgrade Requirements: This release is only updating the code in the plugin itself, so only the Jar is needing to be replaced
- BUG: When the Rest service in question was Remedy, I was hard coding the 'Content-Type' header and not telling you about it. This was needed only back before I allowed custom headers to be define, and the one I was defining for query was fine, but not for create or set, so it was causing issues. I have removed this hard-coded header, and added notification of the addition of the Authentication header that was also always being set
- ENHANCEMENT: Updated the Remedy Sample form that gives examples of Create/Set body templates and finished fully fleshing out the field list, so if you are using this sample form, you will want to re-import the def and re-import the Table and Field data records as well.

2.4 – November 20, 2016

- Upgrade Requirements: There are several supporting JAR replacements, all of the jar's need to be copied to the server and the references updated/added in the pluginsvr_config.xml file appropriately. The templates have been updated for reference.
- BUG: Local Filtering was causing a results list with more entries than there were entries…or in other words, the entries were filtered, but not the results list ☺
- BUG: Local Filtering was causing filtering, but at the wrong level. If the query said give me 1 record (a setfield action) then it was grabbing only the first record of the total resultset, then, if that didn't match the local filter, it returned 0 records. I needed to move the filtering of local before the max return portion of the code

- BUG: Added Try/Catch in the CacheCleanup method to prevent that process from preventing processing properly if there are errors in it
- ENHANCEMENT: Added 'method used' to the debug output just for completeness of detail
- ENHANCEMENT: Added body (template) format validation before making the call
- Added Jackson databind library (used for JSON validation)
- Added Jackson annotations library (used as part of overall Jackson upgrade)
- Updated Jackson core library from version 2.5.4 to 2.8.5

2.3 – February 25, 2016

- Upgrade Requirements: This release is only updating the code in the plugin itself, so only the Jar is needing to be replaced
- ENHANCEMENT: The limitations of the 2.2 release on local filtering were too restrictive for my own needs, so I needed to build out full query capabilities including NOT, >, <, AND, and OR capabilities
- ENHANCEMENT: The query that returns records previously returned all rows and all fields for the record, this was potentially resource intensive on the network/client depending on how many columns are involved. This enhancement makes it so that the fields returned are only those being requested. This will speed up set-field actions as well as table queries by transferring less data between the plugin and the client
- ENHANCEMENT: Previously, the 'control fields' that are controlled by the plugin (Debug, BypassCache, and OutputUnusedElements) were not previously being returned with your search results. Recently I found that when using Vendor forms in Joins, that this caused the records not to display properly., so I changed this process to populate the control fields as well
- BUG: Apparently 9.x changes the way that escalations sends information to the plugin so I needed to move the code around a bit to make the 'Backup' process function properly

2.2 – February 18, 2016

- Upgrade Requirements: There was new workflow and new fields added, so this update requires a re-import of the Remedy code as well as replacement of .jar file
- ENHANCEMENT: Local Filtering, I came across a requirement where the web service I was calling didn't have the ability to give me only the results I wanted (no ability to query on the value), so, in my example they were providing me 50 records, when I actually only wanted to display 10 of them. This lead me to this feature of 'Local Filtering'. The feature is an attribute of the field definition. Read the documentation inline to learn all the details
- ENHANCEMENT: Added the ability to auto-generate fields. After a 'Table' record has been created you can now click the 'Auto Generate Field Records', provide proper sample data and the plugin goes through the request and adds any elements in the resulting call to the Fields list for that table
- ENHANCEMENT: Added the ability to get a Request ID back from the service on a Create function by specifying the element name in 'Create Request ID Element' field
- BUG: When using Cached results, the plugin wasn't storing the results in the resultsMap, so, it was throwing an error when trying to get those results back
- BUG: While the plugin was properly maintaining when to use the previously cached results, and when to get new results, the cache was never being cleaned out, as a consequence, the size of the cache object just kept growing over time taking more memory. I added a mechanism to go through the cache object and clean out expired cache objects
- BUG: It turns out the 'optional' feature I added to templates with 2.1 never actually worked properly (sorry Tony), so, I got that working at this point

2.1 – February 8, 2016

- Upgrade Requirements: To apply this upgrade you will need to import the APL:RestfulVendorForms:Table form to get the new fields, and replace the .jar file on your server
- ENHANCEMENT: With the new template feature, I added the ability to now mark a field as optional in a template. If the field is part of the template, and the field is null, it'll remove the field, and all tags on the line
- ENHANCEMENT: With the addition of CRUD, I hard coded the expected response codes. This is causing issues for some people, so I've made them variables that can be defined as needed
- BUG: I was not checking to ensure that the methods for CRUD were populated, so it's possible that I was going to pass a blank value. I have defaulted the values if nothing is specified in the Table record. CRUD defaults to POST/GET/PUT/DELETE respectively

2.0 – February 2, 2016

- Upgrade Requirements: To upgrade the plugin, you will need the new Jar file as well as importing all of the Remedy Code. This will give you all of the forms/workflow necessary to utilize everything. All of the existing records will continue to work without issue after the import. Additionally, if you want to take advantage of the new Backup RAS Hash feature, that value will need to be added to your pluginsvr_config.xml file.
- ENHANCEMENT: Added the ability to do more than search on a form. With this release you now have the ability to Create, Update, and Delete 'records' on this vendor form.
- ENHANCEMENT: This plugin now supports not just putting things in the URL, but, now also in the body of the request. This is handled through the new 'template' fields defined on the Table form. By providing a mix of hard coded values and field names encased in | values the body will be able to be dynamic based on the contents of the fields. Check out the Template fields above for details about how the field value replacement works
- ENHANCEMENT: Added the ability to add your Remedy Application Service 'hash' from your ar.cfg file instead of a username/password for 'backup' purposes, when the plugin needs to log on and it's the Escalator that's executing the plugin. This feature allows you to store a hashed password without the need to have the ID/PW actually in the configuration file.
- ENHANCEMENT: On the 'Table' form, there is now a button to show you the current contents of the Cache
- ENHANCEMENT: When an error occurs, the error message now provides not just the Error code and URL, but also the Message associated with the error, for better troubleshooting help
- BUG: When a query was issued and 0 results were returned, the plugin still generated 1 entry, essentially an empty entry. If the query returns 0 records, but is still a valid call, then 0 entries will be returned now
- BUG: 'Field' = "Value" worked, but "Value" = 'Field' did not work…I made the assumption that everyone was making their qualifications the same way, this is obviously not true…so, both nomenclatures now work properly

1.9 – January 4, 2016

- Upgrade Requirements: Due to changes in fields on the Table form, you'll need to re-import the Remedy Code, and of course the plugin jar file. With the addition of the 'timeout' value, you'll need to update your pluginsvr_config.xml if you wish to provide a default timeout other than 90 seconds

- ENHANCEMENT: Prior to this release, the connection you made to the remote service didn't have a timeout value, which can cause a timeout when making calls to it because Remedy will timeout waiting for a response. So, I have modified the plugin to have a default plugin value of 90 seconds, and allow you to specify the plugin default within the pluginsvr_config.xml file, as well as a new parameter per table, which will require you to re-import the APL:RestfulVendorForms:Table table to get the new field. If you choose to do none of these things, the new default of 90 still applies, you just don't have ability to override it.
- BUG: There are apparently two types of 'Basic' authentication. One that requires authentication to connect to the web host, and a second that utilizes the Authorization header information to not provide authentication to the web server, but instead, to the app in question. This plugin was previously only doing web server authorization, but not doing header authorization. This has now been added to allow for this to function properly
- CHANGE: Removed the 'username' from the result set name. Originally, was planning on storing the results per user that issued a query, but never implemented…so…the username being stored as part of the result set name was confusing and not needed…doesn't affect any functionality, just an internal structure change

1.8 – November 20, 2015

- ENHANCEMENT: A user came to me yesterday with a web service they were trying to consume that used a query parameter of 'request_id#en='. The problem they were having is that my plugin was not encoding the #, and causing them issues. This confused me because I'm doing proper encoding. After looking into it, the use of # in a URL is an 'invalid' practice and shouldn't be done. The # actually creates what is known as a fragmented URL. Everything past the # is supposed to be interpreted by a browser to indicate 'where on a page to load', and isn't supposed to be sent to the server. Because of this, my code was assuming that everything after the # was a reference to a location on the page, and wasn't encoding it. Well, being the vendor is xMatters, a 'known quantity' in the Remedy world, instead of making it so that my plugin won't work with their tool, I decided to ignore the 'standards' of how it's all done, and encode everything properly…but, truly….they shouldn't be using them in their restful api ☺
- BUG: It was discovered by me that if you build everything properly on your Dev server, and then move your Vendor form to your test environment without moving the 'Table' and 'Field' records, that the plugin wasn't handling that situation properly. I corrected the code to provide a nice and informative error message when this situation occurs.

1.7 – November 2, 2015

- ENHANCEMENT: Added a new field value mapping capability to the JSON interpreter. If you have an array with multiple 'records' in it with named values, the existing functionality gives you the 'last' entry in the array as your field values. This change allows you to specify a value in the array and map the field that matches it in the same record…details are above in the field documentation section.

1.6 – October 9, 2015

- **CHANGE**: A User provided me with a sample XML where the value that would be provided for 'Record Indicator' wasn't unique in the XML, in fact, that element name was duplicated within the element itself, which made it not unique which was causing an improper closing of the record when the child element closed. Because of this, I have modified the code so that the record indicator is fully qualified, so instead of just using 'songs', it's now 'ROOT:songs'. This

change will break existing records that are setup, but only requires a simple modification to the 'Record Indicator' field to be fully qualified instead of only the element name
- BUG: In the XML Parser, the element value wasn't being cleared properly on end element, causing extra elements being populated that weren't populated in the source XML

1.5 – September 21, 2015

- ENHANCEMENT: Added the ability to define 'custom headers' for each call, allowing you to control the headers being sent. This in vital/needed for both Jira and Service Now, without it a 400 'Malformed Request' error is returned.
- ENHANCEMENT: I have now included a sample Service Now configuration that a user was kind enough to give me access to in order to troubleshoot his issue
- ENHANCEMENT: Added the ability for each 'table' to have its own cache setting that overrides the default cache size set in the plugin configuration
- BUG: Fixed a small bug with the caching that was causing it to no longer use a cache after the initial cache for a URL expired
- BUG: Fixed a bug in the JSON parser that was causing rows inside of arrays to not be processed as new rows. I'm sure I did this on purpose at some point during development, but can't recall it, didn't make note of it, so, I'm reversing it…the code is now functioning properly with Service Now calls
- BUG: Fixed some references in some of the Sample code that was preventing it from being imported on some older Remedy servers

1.4 – September 4, 2015

- Initial 'publicly available' download
- ENHANCEMENT: Determined that upon 'Error' of one kind or another, the workflow needed to be able to determine that via workflow. To accomplish this, there is a new 'Default' field available on all Vendor forms (created by this plugin) named 'ErrorOutput' that can be added. If the plugin generates an error, then this field will be populated with the contents of that error. Further, if an error is encountered during calling the service, that cached entry will not be used, and a fresh call to the web service will be utilized
- ENHANCEMENT: This plugin utilizes the currently logged in user to log back into Remedy to get information. When an Escalation performs the execution, it doesn't pass valid credentials to the plugin server to be able to log back in and get the information needed. Because of this, Escalations were throwing errors. I have included 2 new plugin parameters 'BackupUserName' and 'BackupUserPass' in the pluginsvr_config.xml file. These parameters are needed only if an Escalation has the possibility of causing a search against a vendor form.
- BUG: Fixed a problem with the caching mechanism, where every time a cache was used, the cache timestamp updated to the time it was used, causing a cached item to stay cached significantly longer than expected depending on use

1.3 – August 25, 2015

- ENHANCEMENT Modified all 'System.out' calls to be log messages allowing for proper logging as configured for the plugins

1.2 – August 21, 2015

- ENHANCEMENT: Updated some of the documentation based on user feedback
- ENHANCEMENT: Updated some of the definitions, you should re-import the 'Remedy Code' Def file to get the updates

- ENHANCEMENT: Updated the OpenWeather sample to include a Console allowing you to consume both services, seeing the difference between the two, and giving an example of how the results can be used
- BUG: Notified about a defect on the 'getEntry' call which was preventing the results from being displayed, corrected

1.1 – August 13, 2015

- ENHANCEMENT: Added ability to consume Remedy 9.x web services, need to re-import 'Remedy Code' def files

1.0 – August 10, 2015

- Initial public pre-release made available for download

Appendix A: Determining Record Indicator Element/Level

Example XML output

```
<weatherdata>
	<name>Colorado Springs</name>
	<forecast>
		<time day="2015-11-20">
			<temperature min="264.46" max="277.83</temperature>
		</time>
		<time day="2015-11-21">
			<temperature min="254.51" max="275.32</temperature>
		</time>
	</forecast>
</weatherdata>
```

Looking through this XML, you can see that the value 'time' is the one that is repeating, and is the element that defines the rest of the values for a given record...so, the record indicator for this XML is weatherdata:forecast:time, or, if you need, you could specify that the recordElementLevel is 2

Now...the same call, but this time with JSON as the formatting

```
{
	"name": "Colorado Springs",
	"list": [
		{
			"temp": {
				"min": 264.46,
				"max": 277.83
			},
		},
		{
			"temp": {
				"min": 254.51,
				"max": 275.32
			}
```

```
                }
        ]
}
```

Because JSON is a bit different and the root element doesn’t have a 'name', the plugin calls the root element ROOT. Now, if we look at this JSON we can see that the record is in the 'list' array, it gives list: [, and then there is a series of {}, {} after it, each one representing a 'record' being returned within that array. So, in this example, ROOT:list is the record indicator and 1 is the recordElementLevel.

Appendix B: Skipping JSON Levels

One of my users came across an issue where the JSON generated resulted in each ‘record’ in the JSON created unique elements because the record id was part of the element name, this caused them to be unable to generate a ‘generic’ element name that could be used. So, I added the ability to skip various levels of the JSON. Skipping it tells the plugin to not include that level of the JSON in the element names utilized, the Example provided looked something like this


```
{
   "PPL000000148645": {
      "ACD Phone Num": null,
      "Accounting Number": null
   }
)
```


In this instance, without the ‘skip levels’ the element name for Accounting Number would be

ROOT:PPL000000148645:Accounting Number

This obviously won’t work when you are using a different record number, so in this case I set the skip level to 1 and the returned element becomes

ROOT:Accounting Number

The Skip JSON Levels field is a comma separated list of levels you want to skip