Glubol REST Server

API Consumer Guide

August 7th, 2020

Basics

Glubol is a server intended to provide maximum access to the Retail Pro 9 plug-in API (PIAPI) or Retail Pro 8 RDA2, and its REST interface builds on that to give developers a simple means of accessing Retail Pro's data. The intention is to provide access to any business object surfaced by the PIAPI or RDA2, and where called for, to give direct read-only access to any table or view used by V9.

Interoperability

A certain amount of effort went into making the REST interface for the V8 and V9 versions of Glubol as similar as possible. Because the customization layers for Retail Pro 8 and Retail Pro 9 are so different, their individual functionality limitations affect the requirements for using Glubol with each version. Given that, the business objects supported by V8 are also supported by V9, and both versions use the V9 attribute names when processing column and filter URI parameters.

There are several places in this document where we discuss the difference between working with Retail Pro 8 and 9. To aid applications that need to know the difference, responses coming from Glubol include the "X-Powered-By" HTTP header set to either "Retail Pro 8" or "Retail Pro 9".

Glubol’s REST Treatment

This document doesn’t go to any length explaining Fielding's dissertation on hypermedia, but it does explain the application of those concepts to an established point of sale system. The assumption is that you already understand the basics of HTTP resources. Mapping CRUD to HTTP methods, Glubol uses POST for creates, GET for reads, PUT for updates, and, of course, DELETE for deletes.

Note that, in the case of PUTs, this API was developed prior to the implementation of RFC 7231 defining the use of commands to apply updates, and instead requires that you send partial data state, similar to the POST operation. This is described in greater detail later.

Glubol observes the Accept HTTP header when constructing the response. The valid values are "application/xml" and "application/json", resulting in output formatted as XML or JSON, respectively.

Depending on the type of request, Glubol supports the use of the following parameters for qualifying the request.

> cols - A comma delimited list of attribute names as they appear in Retail Pro 9.

filters - A semi-colon delimited list of filter specifications. Each filter consists of an attribute name, an operator, and an optional filter value, separated by commas.
sorts - A semi-colon-delimited list of sort specifications. Each sort consists of an attribute name and an optional direction, separated by a comma.
start - The record on which to start returning results during a GET operation.
limit - The maximum number of records to return during a GET operation.

Authentication

Glubol requires that you authenticate first. This is done by passing the username and password in the customer HTTP headers "glu-auth-username" and "glu-auth-password", respectively, and for version 8, "glu-auth-workstation" to pass the workstation number. The credentials are authenticated against Retail Pro. If the credentials are valid, an authentication token is passed back in the HTTP response header "glu-auth-token". All subsequent operations must use this token, passing it back in the request header "glu-auth-token".

Glubol uses TLS/SSL, and certificates are included in the installation, along with the OpenSSL libraries for use with the GTest utility.

HTTP Methods

GET

To read rows from a resource, a GET operation is performed.
Note that, at this time, the focus of the REST interface is on inventory and customers. While Glubol is designed to work an any PIAPI business object in a generic fashion, the customer and inventory resources have been optimized for speed when performing reads.

An example of a read of a list of customers might look like the following:

> http://server/v1/rest/customer?cols="First Name","Last Name","Address1", "Address2","Phone1","ZIP","Cust Id","Active","Store No"&filters="Sbs No",eq,1;"Last Name",lk,"Sm"&start=100&limit=10

Let's pull that apart. I have the machine that the instance of Glubol is running from...

> **http://server**/v1/rest/customer?cols="First Name","Last Name","Address1", "Address2","Phone1","ZIP","Cust Id","Active","Store No"&filters="Sbs No",eq,1;"Last Name",lk,"Sm"&start=100&limit=10

Followed by the version of the interface and the type of interface...

> http://server**/v1/rest**/customer?cols="First Name","Last Name","Address1", "Address2","Phone1","ZIP","Cust Id","Active","Store No"&filters="Sbs No",eq,1;"Last Name",lk,"Sm"&start=100&limit=10

The "v1" is obvious, but why include "rest"? In case there is ever a need to provide other types of interfaces, RPC included. Next is the resource name.

> http://server/v1/rest**/customer**?cols="First Name","Last Name","Address1", "Address2","Phone1","ZIP","Cust Id","Active","Store No"&filters="Sbs No",eq,1;"Last Name",lk,"Sm"&start=100&limit=10

The attributes I want returned...

> http://server/v1/rest/customer?**cols="First Name","Last Name","Address1", "Address2","Phone1","ZIP","Cust Id","Active","Store No"**&filters="Sbs No",eq,1;"Last Name",lk,"Sm"&start=100&limit=10

...and the filters I want to apply in locating the records.

> http://server/v1/rest/customer?cols="First Name","Last Name","Address1", "Address2","Phone1","ZIP","Cust Id","Active","Store No"**&filters="Sbs No",eq,1;"Last Name",lk,"Sm"**&start=100&limit=10

And, out of the records that match my criteria, I'm going to start at row 100 and get up to 10 records.

> http://server/v1/rest/customer?cols="First Name","Last Name","Address1", "Address2","Phone1","ZIP","Cust Id","Active","Store No"&filters="Sbs No",eq,1;"Last Name",lk,"Sm"**&start=100&limit=10**

Basically we're asking for information about every customer in subsidiary #1 whose last name starts with "Sm". Note that the filter on the subsidiary is required when doing a read for any resource, and you have to specify the name of the subsidiary number attribute as it is defined for the resource. (Put another way, Glubol uses the names for the attributes that were defined by Retail Pro for each of their business objects and, unfortunately, those attribute namings are not consistent across business objects.) That would be "Sbs No" for customers (with a space), "Sbs_No" for inventory , and "Subsidiary Number" for invoices, just to give you an idea. Any other filters are optional, but that subsidiary number must be included or you'll get no data back.

Three content formats are supported: XML, JSON, and CSV. These are specified in the request using the Accept HTTP header and one of:

- application/json
- text/json
- application/xml
- text/xml
- application/csv
- text/csv

If we set the Accept header to "application/xml", the response might look like this, shortened for brevity:

```
<Customers><Customer><ATT0 id='First Name'>Aby</ATT0><ATT1
id='Last Name'>Syafiq</ATT1><ATT2 id='Address1'>399 Burj
Blvd</ATT2><ATT3 id='Address2'>Dubai, UAE</ATT3><ATT4
id='Phone1'>789-3213</ATT4><ATT5 id='ZIP'>526-A685</ATT5><ATT6
id='Cust Id'>115</ATT6><ATT7 id='Active'>1</ATT7><ATT8 id='Store
No'>1</ATT8></Customer><Customer>...
</Customers>
```

If you're used to seeing the tags echo the field name, this might look unusual. The reason for using "id" XML attributes is because V9 attribute names often contain spaces and special characters. This makes it impossible to use them as XML tag names without modifying them. I could have come up with some rules for normalizing the attribute names, but past experience says that most normalization results in naming conflicts. The easiest solution was to name the fields ATTx where x is the position of the attribute in the request and put the name of the attribute in the XML attribute "id".

If no records match the filter specification, you do NOT get a 404 not found. That is reserved for situations in which the resource name is incorrect. In other words, a 404 indicates that the specified resource container wasn't found.

Retail Pro 9 Optimizations

In Glubol for Retail Pro 9, some of the resources have been optimized for speed by bypassing the PIAPI and building SQL statements using the request criteria. The customer, inventory, invoice, and slip resources have been optimized in this manner. Optimization for the remaining top level business objects can be done upon request.

POST

A request to create a new record must specify the Content-Type HTTP header to indicate the format of the post body used as the payload. Bulk inserts are supported, in which multiple records can be inserted in one operation.

The URL for an insert to the customer resource would look like the following:

```
http://server/v1/rest/customer?filter="Sbs No",eq,1
```

The filter might seem odd when doing an insert. The reason it's shown in this example is because, when working with Retail Pro 9, the PIAPI requires that ***everything*** be filtered by subsidiary number. You can't do anything with any business objects until you've told it what subsidiary you're working with. That restriction is not imposed by the PIAPI -- it's how the business objects work that the PIAPI gives you access to.

When working with Retail Pro 8, the filter isn't needed and is ignored.

If we set the Content-Type to "text/json", the new customer might look like:

```
Customer:[{"First Name":"Aby","Last Name":"Syafiq","Address1":"399 Burj Blvd","Address2":"Dubai, UAE","Phone1":"789-3213","ZIP":"526-A685","Active":"1","Store No":"1"}]
```

When a record is successfully inserted, the identifying attribute value is returned using the same format as the insert payload. In this case, that might look like:

```
Customer:[{"Cust Sid":"89035689218"}]
```

And the Content-Type would be set to "application/json".

PUT

As mentioned earlier, the logic for PUTs was developed prior to the implementation of RFC 7231, so it requires that you specify only the data state to apply. Any columns left out are ignored.

Updating a record in a resource requires identifying the individual record and supplying the values to replace. Values not specified in the payload are not altered. A URL for updating the customer in the previous example to change Mr. Syafiq's phone number would look like this:

```
 http://server/v1/rest/customer?filter="Sbs No",eq,1;"Cust
Sid",eq,"89035689218"
```

And the payload would look like this, if we were using XML:

```
<Customers><Customer><ATT0 id="Phone1">876-5432</ATT0> </
Customer></Customers>
```

If the customer can't be located -- if the customer SID doesn't exist in subsidiary #1 -- you can expect to get a 404. In the case of a PUT operation, the focus isn't the resource container, it's a single customer resource -- a single record -- and if that record isn't found, the operation can't proceed.

Because V9 doesn't have modified date/time attributes on all records, it's not possible to avoid conflicts on all resources, but it is highly encouraged that you use the modified date/time, where possible as one of the filter values. That way, if the record is not found, one of two things occurred; either the record was deleted or the record was modified. If the record was modified, you can make a decision on whether to overwrite those changes.

Note that PUT is applied only to the first record that matches the filter criteria. If you don't supply a filter that resolves to a single record, the changes are applied to the first matching record only.

DELETE

The URL for a delete operation looks a lot like one for an update operation. You need to identify the record in question. Working with the previous example, if we decide to delete Mr. Syafiq's customer record, the URL would look like:

```
http://server/v1/rest/customer?filter="Sbs No",eq,1;"Cust
Sid",eq,"89035689218"
```

The HTTP method would be DELETE. There is no response payload to a delete request; the only response is the HTTP response code.

HTTP Response Codes

The following are the HTTP codes you can expect to get back from the operations described above.

HTTP Code	Description	Context
200	Ok	Indicates a successful authentication, read, update, or delete.
201	Created	Indicates a successful insert.
400	Bad request	Something about the request was invalid -- details are included in the response body and, optionally, in the log.
401	Unauthorized	An operation was attempted without including a valid auth token.
403	Forbidden	An operation was attempted that was rejected by V9 based on permissions.
404	Not found	The resource could not be located or the container name is not supported.
405	Method not allowed	The HTTP method is not allowed with the specified resource, or is not supported at all.
500	Internal server error	An error occurred in the server that is unrelated to the request. Details are included in the response body and, optionally, in the log.

Resources

The next few sections describe the resources available to the developer when constructing URLs and payloads. In this version of this document, only the customer and inventory are described in detail. The rest will be included in subsequent updates to the software and this documentation.

NOTE: These tables are not complete. While I've attempted to list some of the fields and their uses, GTest should be considered the best reference for supported fields.

Customer

(In subsequent versions of the documentation, I'll fill in the descriptions. For now, I just wanted to get the attribute names listed.)

Attribute name	PUT/ POST	Description
Accept Checks		Can this customer use checks when tendering?
Active	x	Is this customer active?
Addr Type Id		Identifier for address type record
Address1	X	The customer's primary mailing address
Address2	X	
Address3	X	
Address4	X	
Address5	X	
Address6	X	
Allow Email		Customer may be emailed
Allow Phone		Customer may be called
Allow Post		Letters may be mailed to customer
Archived		Customer record has been archived
Begin Date (stat)		
Charge Balance		Customer's charge balance
Charge Limit		Maximum amount when using charge as tender type
Check Limit		Maximum amount customer may tender by check (cumulative)
Company Id		Company the customer represents
Company		
Controller		Identifies the database to which the customer record currently belongs
Country		Country portion of mailing address.
Country Id		
Created By		The username of the employee that created this customer record
Created By Id		The employee ID of the employee that created this customer record
Created Date		The date and time this customer was created (local time zone assumed)
Cust Class		Customer class
Cust Class Id		
Cust Sec Lvl		Customer security level
Cust Sid		Unique identifier across enterprise
Cust Type		Customer type. 0=Local, 1=Global, 2=Regional, 3=Not shared
Default Addr No		Indicates which address is the default.
Discount Perc	x	Customer discount percentage
EMail	x	Customer's primary email address
End Date (stat)		

Attribute name	PUT/ POST	Description
Exported to AR		Has this customer been exported to the accounts receivable system?
First Name	x	
Full_Name_Search		
Household Code		
Info1		Legacy user defined data entry fields.
Info2		
Last Name	X	
Last Sale Date	X	
Mark1		Accounting system flags
Mark2		
Max Discount Perc		Maximum discount percent that can be applied to this customer. Default is derived from system configuration.
Modified By		The username of the employee that last modified the balance information on this customer record.
Modified By Id		The employee ID of the employee that last modified the balance information on this customer record.
Modified Date	x	Date and time the balance information on this customer was last modified (local time zone assumed)
Notes		Free text
Orig Controller		Identifies database from which the customer record originated.
Phone1	X	Phone 1 from the customer's primary address
Phone2	X	Phone 2 from the customer's primary address
Price Level	x	Starting price level override for this customer. Default is null.
Primary Associate		Employee name for the primary sales clerk for this customer.
Primary Clerk		Employee ID for the primary sales clerk for this customer.
Priority		Shipping priority
Region		Region in which this customer resides / shops.
Region Id		
Sector		Sector in which this customer resides / shops.
Sector Id		
Security Level		Security level associated with this customer. Zero or null implies no security restrictions.
Station		Legacy field
Status		Customer GL status: 0=Changed, 1=Transmit, 2=Set, 3=Ignored 4=Deleted
Store Code		Customer's home store
Store No	X	
Subsidiary		Customer's home subsidiary
Tax Area		Tax area override that applies to this customer
Tax Area 2		
Tax Area Id		
Tax Area Id 2		
Term Type		Customer payment terms.
UDF1 Date		User defined dates.
UDF2 Date		
Zip	X	Postal code

Inventory

Attribute name	PUT/ POST	Description
ALU	x	Alternate Look Up
Attr	X	Attribute (some quality other than size)
Catalog Id		Web store catalog ID
Cmp Max Qty		Company maximum quantity
Cmp Min Qty		Company minimum quantity
Cmp OH Qty		Company on-hand quantity
Cmp OnOrder Qty		Company on-order quantity. Populated by SRO process.
Cmp Rcvd Qty		Company received quantity. Populated by SRO process.
Cmp Sold Qty		Company sold quantity. Populated by SRO process.
Cmp Total Qty		Company total quantity
Cmp Transfer In Qty		Company transfer in quantity
Cmp Transfer Out Qty		Company transfer out quantity
Commission Code Id	X	Commission code
Corporate	X	Did this inventory item originate in a V9 system?
Cost	X	
Created Date	X	Date this inventory record was created (local time zone assumed)
Currency ID	X	Identifier of currency used to express monetary values on this record
DCS Code	X	Department / Class / Subclass
Description1	X	Description of inventory item.
Description2	X	
Description3	X	
Description4	X	
Discountinued Date	X	Date inventory item was discontinued.
EC Flag	X	ECI flag. Will this item be communicated to a web store?

Attribute name	PUT/ POST	Description
EXT_FLAG	X	Bitmask. Contains lot and serial number switches.
First Received Date		Date inventory item was first received.
FLAG	X	Bitmask. Contains flags governing inventory workflow.
Foreign Order Cost	X	
Gift Flag	X	
Height	X	
Item No		Legacy inventory item identifier.
Item Note	X	Free text
Item Sid		Unique identifier. Used to associate other records with this record.
Item Status		Item state. 0=regular, 1=proposed, 2=rejected
KeyItem Group Id	X	
Kit Type		Bitmask. Contains flags indicating type of kit or package.
Last Mkdn		Date of last markdown price adjustment
Last Received Cost	X	Cost per unit of inventory item when last received
Last Received Date		Date inventory item was last received
Last Sold Date		Date quantity of inventory item was last sold
Length		
MasterSbs		
Max Disc Percent 1	X	Maximum discount percentages for this inventory item
Max Disc Percent 2	X	
Min Order Qty	X	Minimum quantity to use when ordering this inventory item
MinMax End Date		Min/Max date range
MinMax Start Date		
Modified Date		Date and time this inventory item record was last modified (local time zone assumed)
New Max Price		
New Max Qty		
New Min Price		
New Min Qty		
Orderable Date	X	
Oversized Item	X	Is this inventory oversized and require special shipping?
Print Tag Flag	X	Used to flag inventory items for tag printing
Prod Cost	X	Production cost
Promotion No	X	Promotional pricing schedule number
QtyPerCase	X	Number of items per case
Range Id	X	
RegionalItem	X	Is this a regional inventory item
SbsNo	X	Subsidiary number
Scale Attr Ord		
Scale No	X	
Scale Size Ord		
Sellable Date	X	Date after which the item may be sold
Ship Method		User defined class of shipping service used for this item
Ship Weight1	X	Weight, including packaging, to use when using ground shipping
Ship Weight2	X	Weight, including packaging, to use when using with expedited shipping
Sid Source		Indicates method of item SID generation
Size	X	
Stored Frmr Price	X	Mark down price
stored_first_price	X	First price assigned to the item
Stored_Price		Current price
Str Max Qty		Maximum quantity to keep on hand
Str Min Qty		Minimum quantity to keep on hand
Str OH Qty		Store on-hand quantity

Attribute name	PUT/ POST	Description
Str OnOrder Qty		Quantity currently on order
Str Rcvd Qty		Quantity received
Str Sold Qty		Quantity sold
Str Transfer In Qty		Quantity transferred in
Str Transfer Out Qty		Quantity transferred out
Style Sid	X	Identifier for style to which this inventory item belongs
StyleDef		
Sublocation Flag	X	
Subsidiary		Subsidiary name
Tag Design		Indicates which tag print layout to use
Tag ID	X	
Tax Code	X	
Text1	X	Free text
Text10	X	
Text2	X	
Text3	X	
Text4	X	
Text5	X	
Text6	X	
Text7	X	
Text8	X	
Text9	X	
Trade Discount Percent	X	
UDF Date	X	User defined fields
UDF Name	X	
Unorderable Flag	X	Indicates inventory item may not be ordered
UPC	X	Universal Product Code
Vendor Code	X	Vendor code
Vendor Lead Time	X	Lead time to use when ordering items from vendor
Vendor List Cost	X	Vendor's list cost for this item
Width	X	

Inventory Quantities

Attribute name	PUT/ POST	Description
ITEM_SID	X	
STORE_NAME		
SBS_NO	X	
MIN_QTY	X	
MAX_QTY	X	
TRANSFER_IN_QTY	X	
TRANSFER_OUT_QTY	X	
QTY	X	
Store Code		
Sold_Quantity	X	
Rcvd_qty	X	
ActiveStore	X	
ASN_IN	X	
Store No	X	
Subsidiary Name		
Committed IN	X	

Inventory Prices

Attribute name	PUT/ POST	Description
ITEM_SID	x	
SBS_NO	X	
SEASON_ID	X	
PRICE_LVL	X	
StoredPrice	X	
Qty Req	X	
Price Level		
Season		
Tax Perc		
Price		
PriceWT		
Tax Code		
Secured		

Invoice

Attribute name	Description

(to be completed -- please refer to GTest for a complete list)

Invoice Item

Attribute name	Description

(to be completed -- please refer to GTest for a complete list)

Tender

Attribute name	Description

(to be completed -- please refer to GTest for a complete list)

Slip

Attribute name	Description

(to be completed -- please refer to GTest for a complete list)

Slip Item

Attribute name	Description

(to be completed -- please refer to GTest for a complete list)

Sales Order

Attribute name	Description

(to be completed -- please refer to GTest for a complete list)

Sales Order Item

Attribute name	Description

(to be completed -- please refer to GTest for a complete list)

Sales Order Deposit

Attribute name	Description

(to be completed -- please refer to GTest for a complete list)

Purchase Order

Attribute name	Description

(to be completed -- please refer to GTest for a complete list)

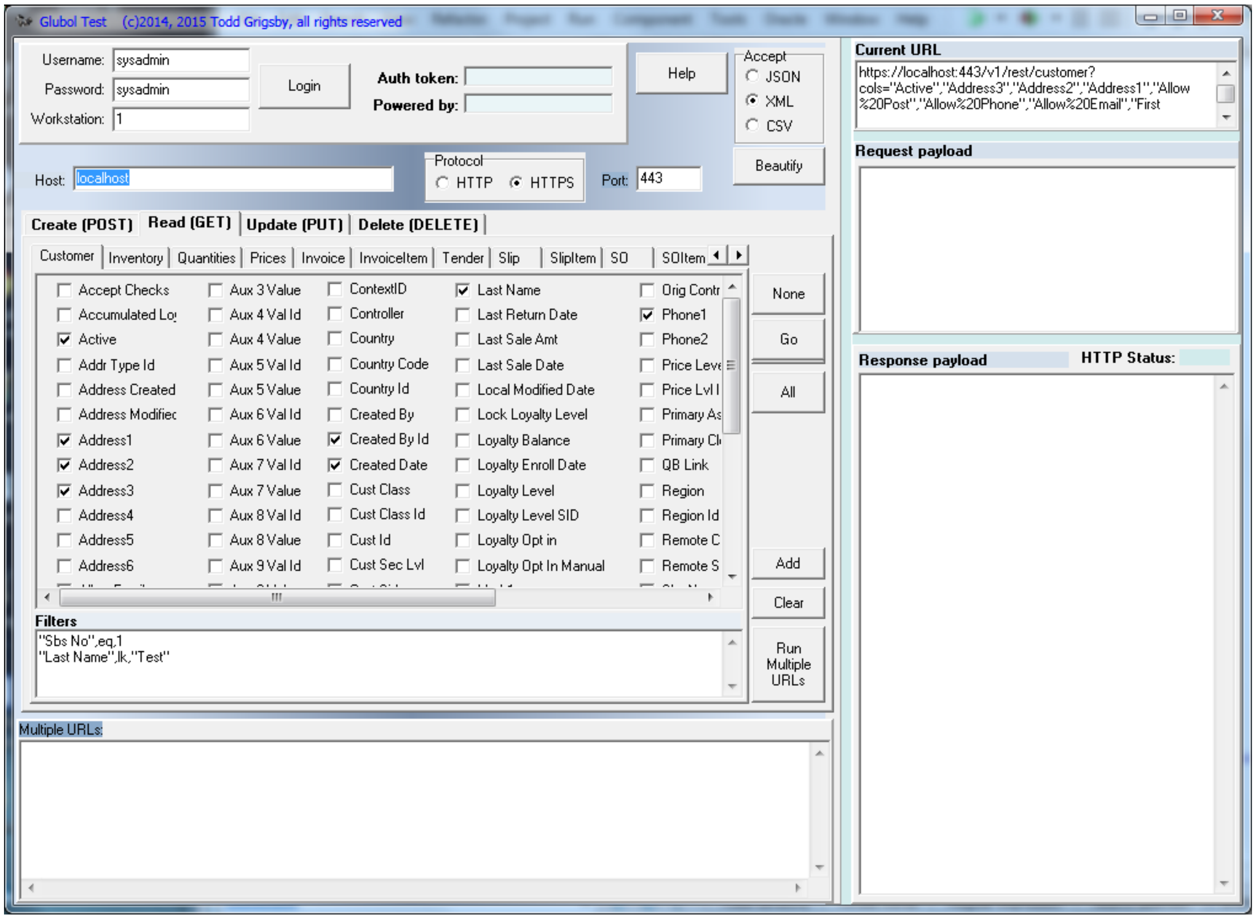
Purchase Order Item

Attribute name	Description

(to be completed -- please refer to GTest for a complete list)

Testing with GTest

The GTest utility was designed to make it easy to try out the Glubol REST interface and understand how it works. Let's start with a screen shot.



Username: sysadmin
Password: sysadmin
Workstation: 1
Auth token:
Powered by:
Accept: JSON [✓ XML] CSV
Host: localhost
Protocol: HTTP [✓ HTTPS]
Port: 443

Accept Checks	Aux 3 Value	ContextID	✓ Last Name	Orig Contr
Accumulated Lo	Aux 4 Val Id	Controller	Last Return Date	✓ Phone1
✓ Active	Aux 4 Value	Country	Last Sale Amt	Phone2
Addr Type Id	Aux 5 Val Id	Country Code	Last Sale Date	Price Leve
Address Created	Aux 5 Value	Country Id	Local Modified Date	Price Lvl I
Address Modifiec	Aux 6 Val Id	Created By	Lock Loyalty Level	Primary As
✓ Address1	Aux 6 Value	✓ Created By Id	Loyalty Balance	Primary Cl
✓ Address2	Aux 7 Val Id	✓ Created Date	Loyalty Enroll Date	QB Link
✓ Address3	Aux 7 Value	Cust Class	Loyalty Level	Region
Address4	Aux 8 Val Id	Cust Class Id	Loyalty Level SID	Region Id
Address5	Aux 8 Value	Cust Id	Loyalty Opt in	Remote C
Address6	Aux 9 Val Id	Cust Sec Lvl	Loyalty Opt In Manual	Remote S

Filters:
"Sbs No",eq,1
"Last Name",lk,"Test"

Current URL:
https://localhost:443/v1/rest/customer?cols="Active","Address3","Address2","Address1","Allow%20Post","Allow%20Phone","Allow%20Email","First

The Help button is useful for getting started but not terribly informative.

Start-Up

When you start up the utility, the first thing you'll be concerned with is logging into the server. Check the host to make sure you're pointing to the correct server. Also check the port; Glubol defaults to port 443, the standard port for TLS/SSL connections, but it may have been configured for a different port. Then set the username and password and click "Login". If the operation was successful you'll receive an authentication token which will display in the "Auth token" field. You won't see the entire thing -- it's pretty long -- but the important thing is that you got a value back and not an error.   Also, in the "Powered By" field, you'll see the value of the "X-Powered-By" HTTP header returned by Glubol, which explains which version of Glubol you're talking to.

Setting Up a Read Request

We'll start with a simple read request, pulling data back from the server. You'll use the notebook control labeled "Attributes" to select a resource using the tabs across the top and the attributes in that resource by toggling the checkboxes next to each attribute. As you modify the request, you'll see the contents of the "Current URL" memo box change. What you're seeing is the actual request that will be sent to the server, minus the request payload.

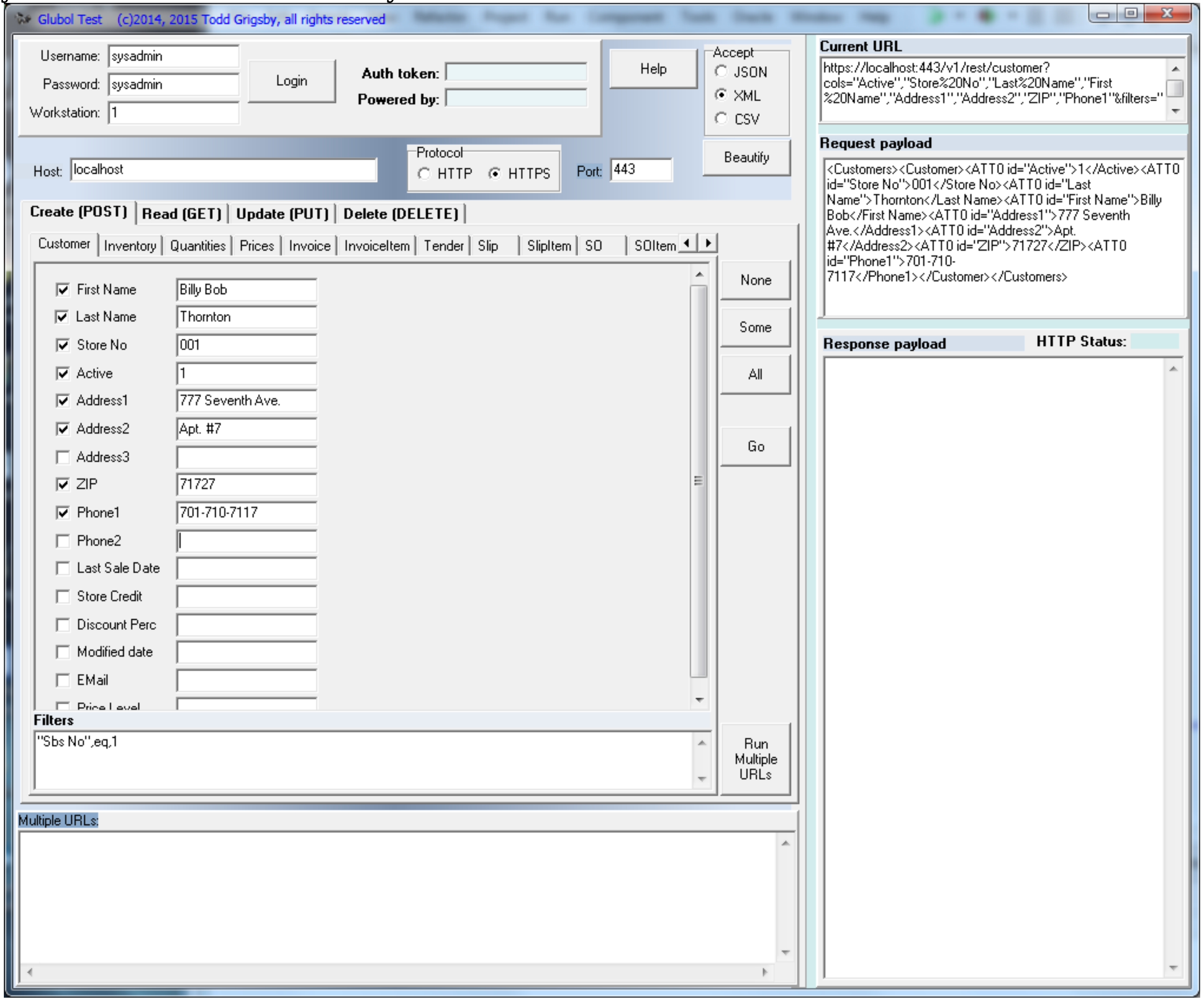
Once you've select the resource and checked attributes you're interested in retrieving, you can create filters for the request. Keep in mind that nearly every business object accessible through Retail Pro 9's plug-in API requires a filter on the subsidiary number. The name of the subsidiary number attribute often differs, so be careful how you set that up. Each filter is placed on its own line in the control. Again, as you change the filters, the changes appear in the "Current URL" at the bottom.

Once you have the request set up, click "Go". The request is submitted to the server and the response is displayed in the "Response Payload" memo control, with the resulting HTTP code displayed in the field labeled "HTTP Status".

If you selected XML, rather than JSON or CSV, as the response format, you can use the "Beautify" button to format the XML for easier reading. Note that it doesn't do a great job -- we'll get back to that when we have time...

Setting Up a Create Request

To create a new record, click on the "Create (POST)" tab, then select the resource you wish to add a new entry for.



Username: sysadmin
Password: sysadmin
Workstation: 1
Auth token:
Powered by:
Accept: JSON [✓ XML] CSV
Host: localhost
Protocol: HTTP [✓ HTTPS]
Port: 443

✓ First Name	Billy Bob
✓ Last Name	Thornton
✓ Store No	001
✓ Active	1
✓ Address1	777 Seventh Ave.
✓ Address2	Apt. #7
☐ Address3	
✓ ZIP	71727
✓ Phone1	701-710-7117
☐ Phone2	
☐ Last Sale Date	
☐ Store Credit	
☐ Discount Perc	
☐ Modified date	
☐ EMail	
☐ Price Level	

Filters
```
"Sbs No",eq,1
```

Current URL
```
https://localhost:443/v1/rest/customer?
cols="Active","Store%20No","Last%20Name","First
%20Name","Address1","Address2","ZIP","Phone1"&filters="
```

Request payload
```
<Customers><Customer><ATT0 id="Active">1</Active><ATT0
id="Store No">001</Store No><ATT0 id="Last
Name">Thornton</Last Name><ATT0 id="First Name">Billy
Bob</First Name><ATT0 id="Address1">777 Seventh
Ave.</Address1><ATT0 id="Address2">Apt.
#7</Address2><ATT0 id="ZIP">71727</ZIP><ATT0
id="Phone1">701-710-
7117</Phone1></Customer></Customers>
```

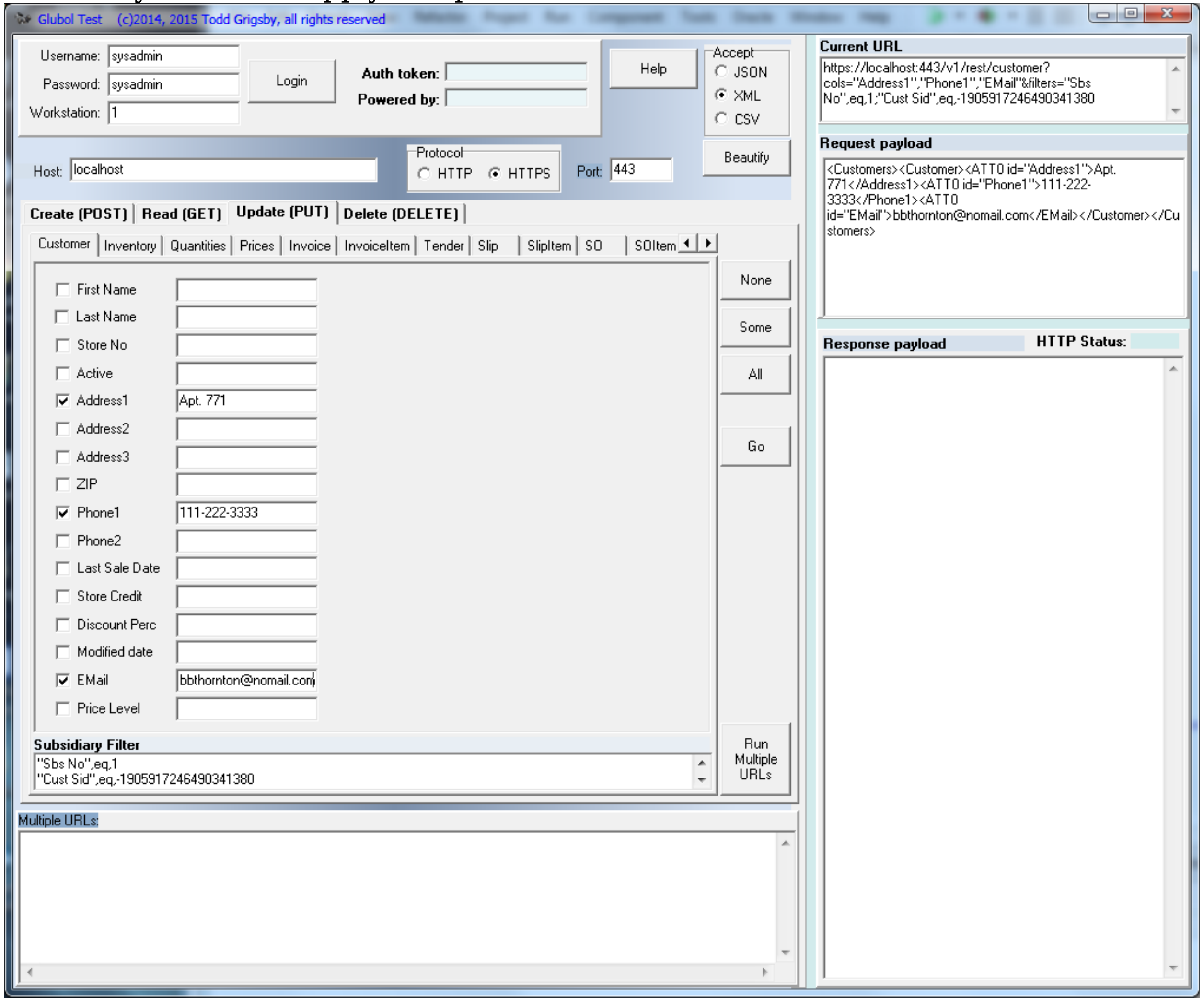
The Create tabs are all set up with check marks and edit fields for the values. To include a column, check the box next to it, then supply a value. If a column is checked and no value is specified, the assumption is that you want to put an empty value in that column. Notice that the filter list must include a filter on the subsidiary when using Retail Pro 9. Glubol for Retail Pro 8 will ignore this filter.

Be aware that no data type or format checking is done by GTest. It will attempt to send whatever value you pass in. And keep in mind that Retail Pro business rules apply -- if the data passed in results in an invalid data state, Retail Pro will reject the record and Glubol will pass back an error.

When you've constructed the record to insert, click the "Go" button. The request payload will be transmitted as the POST body using the URL shown in "Current URL". The HTTP status code will be displayed immediately above the response payload.

Setting Up an Update Request

To modify an existing record, click on the "Update (PUT)" tab, then select the resource you wish to apply an update to.



Glubol Test (c)2014, 2015 Todd Grigsby, all rights reserved

Username: sysadmin
Password: sysadmin
Workstation: 1
Auth token:
Powered by:
Accept: JSON [✓ XML] CSV
Host: localhost
Protocol: HTTP [✓ HTTPS]
Port: 443

Create (POST) | Read (GET) | Update (PUT) | Delete (DELETE)

Customer | Inventory | Quantities | Prices | Invoice | InvoiceItem | Tender | Slip | SlipItem | SO | SOItem

☐ First Name	
☐ Last Name	
☐ Store No	
☐ Active	
☑ Address1	Apt. 771
☐ Address2	
☐ Address3	
☐ ZIP	
☑ Phone1	111-222-3333
☐ Phone2	
☐ Last Sale Date	
☐ Store Credit	
☐ Discount Perc	
☐ Modified date	
☑ EMail	bbthornton@nomail.com
☐ Price Level	

Subsidiary Filter

```
"Sbs No",eq,1
"Cust Sid",eq,-1905917246490341380
```

Multiple URLs:

Current URL

```
https://localhost:443/v1/rest/customer?cols="Address1","Phone1","EMail"&filters="Sbs No",eq,1;"Cust Sid",eq,-1905917246490341380
```

Request payload

```
<Customers><Customer><ATT0 id="Address1">Apt. 771</Address1><ATT0 id="Phone1">111-222-3333</Phone1><ATT0 id="EMail">bbthornton@nomail.com</EMail></Customer></Customers>
```

Response payload

HTTP Status:

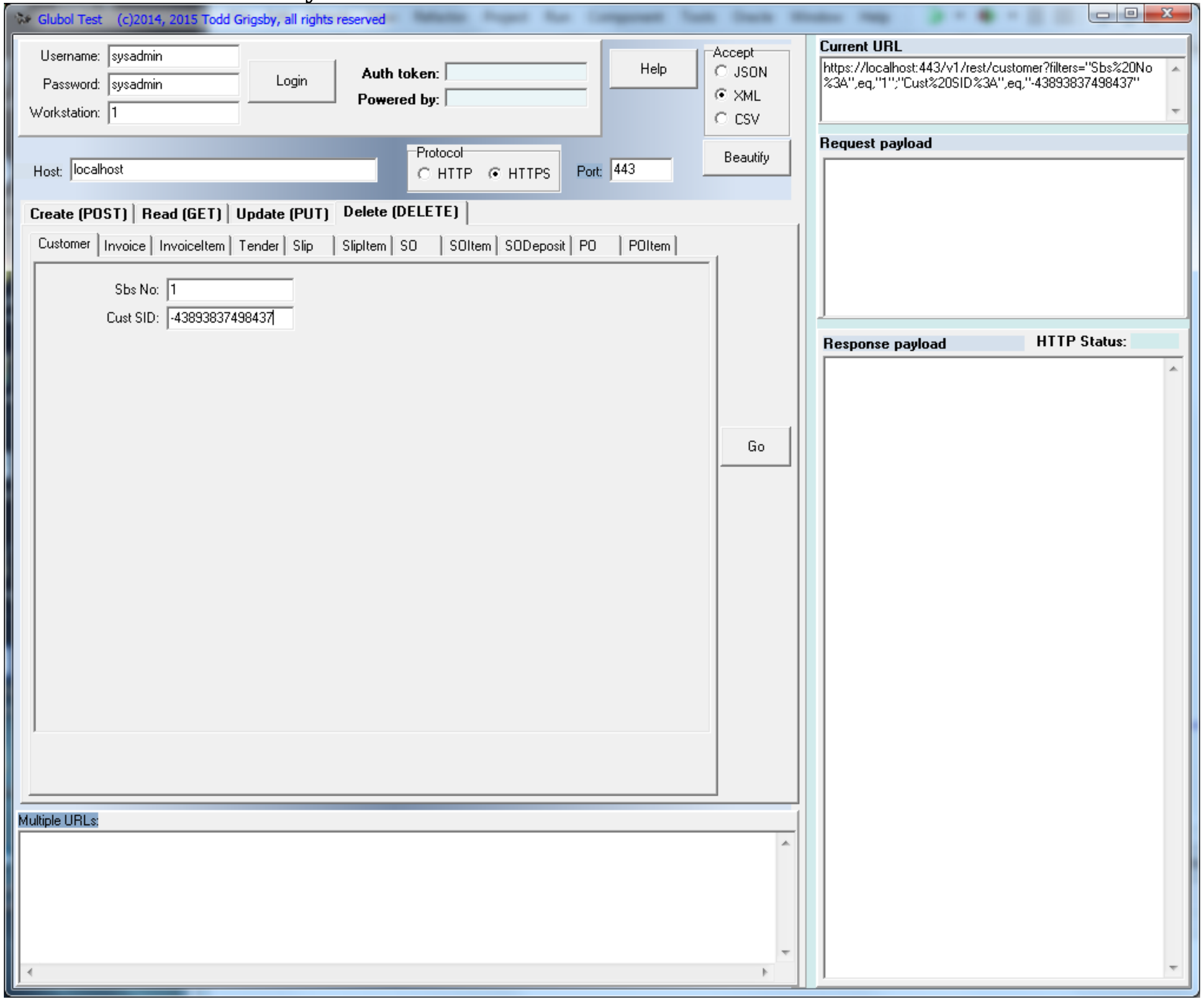
The filter must include the information necessary to identify a single record. Glubol for Retail Pro 9 will need an additional filter on the subsidiary.

Each column checked represents a column you wish to alter in the target record. If you check a column and leave the value blank, you're telling Glubol you wish to empty that field in the record. And again, Retail Pro business rules apply. Some fields are read only and cannot be updated -- and where possible, these are not included in the list of fields in Glubol. And in some cases, the fields are writable but must contain valid values. If Retail Pro doesn't like the data state, Glubol will pass back an error.

When you've finished specifying which fields to alter, click the "Go" button. The request payload will be transmitted as the PUT body using the URL shown in "Current URL". The HTTP status code will be displayed immediately above the response payload.

Setting Up a Delete Request

To delete an existing record, click on the "Delete (DELETE)" tab, then select the resource from which you wish to delete a record.



Username: sysadmin
Password: sysadmin
Workstation: 1
Accept: XML
Host: localhost
Protocol: HTTPS
Port: 443
Sbs No: 1
Cust SID: -43893837498437

Current URL
https://localhost:443/v1/rest/customer?filters="Sbs%20No%3A",eq,"1";"Cust%20SID%3A",eq,"-43893837498437"

You must specify the information necessary to identify a single record to delete. Retail Pro 9 requires a value for the subsidiary; Glubol for Retail Pro 8 ignores that filter.

Once you've identified the record in question, click the "Go" button. The HTTP status code will be displayed immediately above the response payload. Note that not all records can be deleted. Retail Pro's business rules apply, and if a record cannot be deleted due to data integrity rules, Glubol will return a "409 Conflict" error.

Performance Testing

The "Multiple URLs" memo control can be used to load up any number of requests so they can be executed one at a time. Note that, right now, this only works with read requests -- work is being done to expand this to create, update, and delete operations, but it's not ready yet.

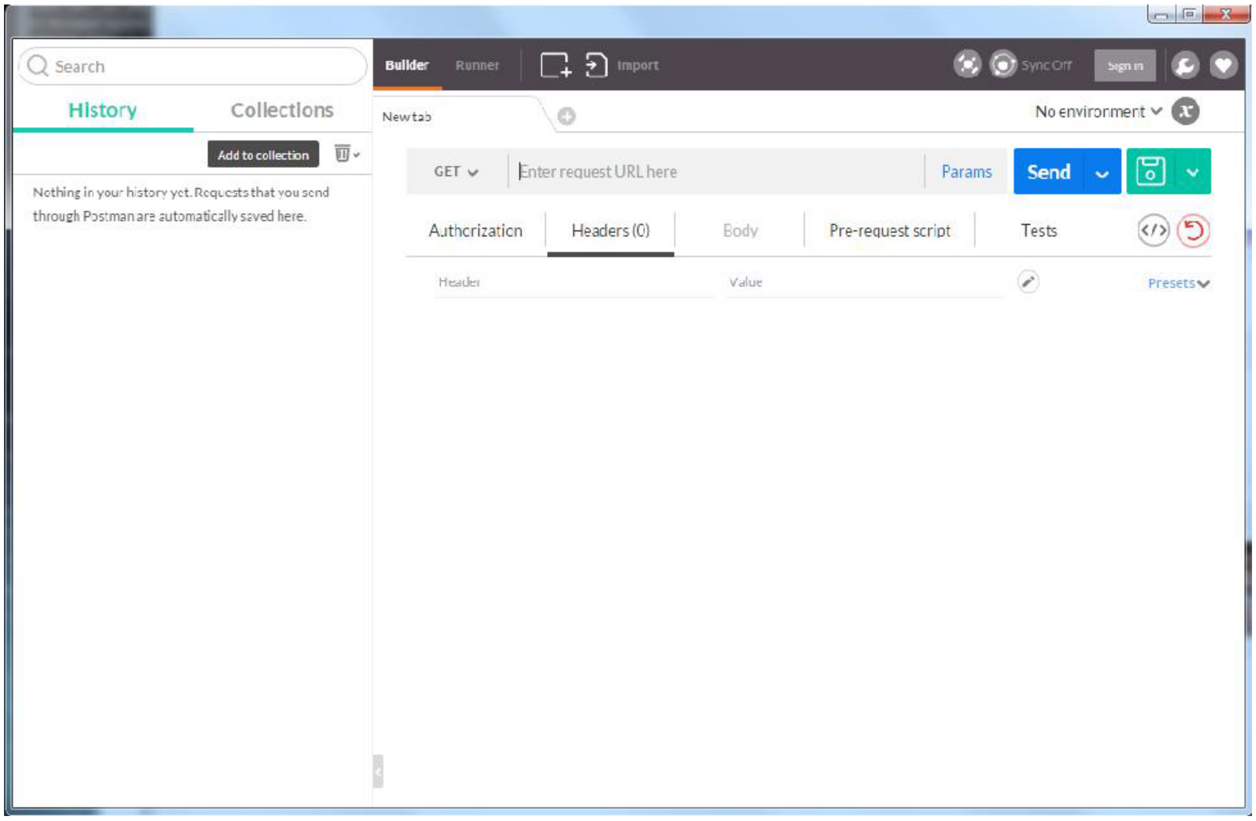
Once you've constructed a GET request, you can add it to the "Multiple URLs" memo by clicking the "Add" button. Alternatively, you can paste a list of requests into the memo field. Clicking the button labeled "Multiple URLs" then executes the requests, one transaction per line. This can be useful for testing the speed of the server.

My recommendation is that, if you're going to test the server in this fashion, that you run multiple instances of GTest and run multiple requests from each instance. In stress testing the server, I will run 5 instances of GTest, each with 100 requests, start them all up and time them. No, it's not terribly accurate, but it's a good way to get a feel for the responsiveness and scalability of Glubol.

Testing with Postman

Postman is a Google Chrome app created by developers for developers. From the app's Chrome extensions page, Postman helps you build, test, and document APIs. It has a number of features that are useful to the web application developer.

Once you've downloaded, installed, and started up Postman, you should see something similar to the following screen:



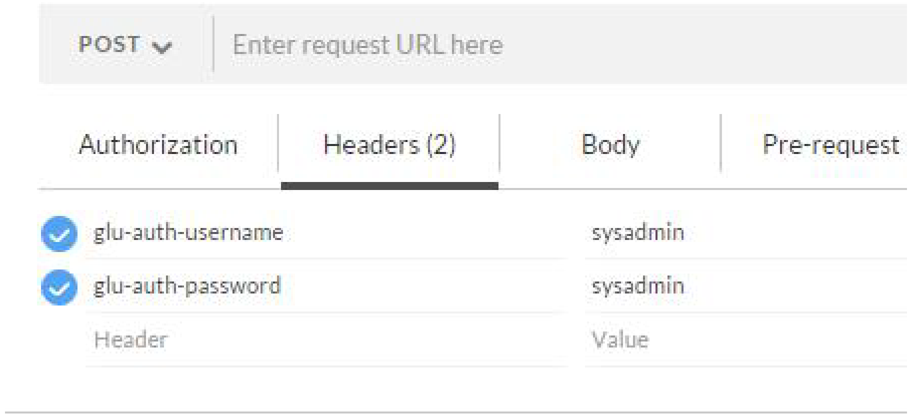
Nothing in your history yet. Requests that you send through Postman are automatically saved here.

A detailed exploration of this extension will be left as an exercise for the reader. For now we'll focus on how to use this extension when interacting with all flavors of Glubol.

Authentication

You first have to authenticate against the server. To do this:

1. Select "POST" as the operation instead of "GET"
2. In the "Enter request URL here" control, enter the following, replace localhost with the name/IP of the server if necessary: https://localhost/v1/rest/auth
3. In the request headers area, enter the following name-value pairs:
    a. glu-auth-username - sysadmin
    b. glu-auth-password - sysadmin
4. When you're ready to post, it should look like this:



glu-auth-username	sysadmin
glu-auth-password	sysadmin
Header	Value

5. Click "Send".
6. If Glubol successfully authenticates you, the status will return with "200 OK". There is no response body.
7. Click "Headers". There are four response headers you're interested in.
    a. Allow. This will list the HTTP verbs that can be used with this flavor of Glubol.
    b. Server. This will be set to a text value that represents the flavor of Glubol. The possible values at this time are GLUBOLV8, GLUBOLV9, and GLUBOLLITE.
    c. X-Powered-By. This value reflects the Retail Pro product that the Glubol server is associated with. At this time the possible values are "Retail Pro 8" or "Retail Pro 9".
    d. glu-auth-token. This value is important. All other requests you make will need to include this header name-value pair.
8. Copy the glu-auth-token value.

Reading a Resource

Now that we're authenticated, let's read data from a resource. In this example, we're going to ask for all
customers whose first names contain an "S". Note that, in the filter parameter, we include "Sbs No",eq,1. Queries against a resource always require a filter on the subsidiary.

1. Select "GET" as the operation to perform.
2. Enter the following into URL, replacing localhost with the IP/name of the server if necessary: http://localhost/v1/rest/customer.
3. In the request headers area, enter "glu-auth-token" and the token value you copied as the last step of authenticating.
4. Click "Params" to display an area for entering URL params, and enter the following keys and values. Notice that we're using URL encoding for spaces and special characters in the column names.
    a. cols -
    b. "First%20Name","Last%20Name","Phone1","EMail","Cust %20Type","Cust%20Sid","Created%20Date","Created%20By%20Id"
    c. filters - "Sbs No",eq,1;"First Name",lk,"S"
5. Click "Send".
6. How long it takes to return depends on how many records you have in your database. The default limit, though, is 100 rows. If you want to set that lower or higher, or start on a row other than the first matching row, see the notes at the top of this document for the URL parameters "start" and "limit".