

IVR Interface Implementation Guide

New World ERP



©2018 Tyler Technologies, Inc.

Data used to illustrate the reports and screens may include names of individuals, companies, brands, and products. All of these names are fictitious; any similarities to actual names are entirely coincidental. Further, any illustrations of report formats or screen images are examples only, and reflect how a typical customer would install and use the product.

Contents

IVR Interface Implementation Guide	1
Introduction	1
How to Use This Guide	1
Setup and Configuration	2
Application Setup	2
Taking in Payments from the IVR	2
Virtual Payment Source	2
Presenting Information to the Consumer	6
Web API Test Client	8
Handoff	10
Troubleshooting	10

IVR INTERFACE IMPLEMENTATION GUIDE

INTRODUCTION

The Interactive Voice Response Interface enables the New World ERP Utility Management system to work with a third party IVR system. Basic account and account history information is fed to the IVR for presentation to the customer. Payments are fed from the IVR to New World ERP Receipt Processing.

This is accomplished with an existing standard interface that requires no customer software development by Tyler. However, the third party system will need some customization to meet New World ERP standards.

HOW TO USE THIS GUIDE

This guide is composed of two main sections. The first section describes what must be done to prepare for the use of the interface. The second section describes some considerations regarding the day-to-day use of the interface.

More specifically, this guide consists of the following sections:

- Application Setup. This section describes the setup required to do the following: 1) take in payments from the IVR; 2) create the virtual payment source in which the IVR batches are defined and configured; 3) present information to the consumer.
- Web API Test Client. This section describes how to test the interface to ensure that it is installed properly. Testing is optional.
- Troubleshooting. This section describes how to utilize system logs to view information about any setup errors, application failures, and/or failed login attempts.

SETUP AND CONFIGURATION

APPLICATION SETUP

Note: The web API service is a separate line item in your license key. Verify in Management Console that you have the 'ivr interface ut' item activated under 'Utility Management' prior to configuring the New World ERP application.

Payments processed through the IVR will be fed into the standard Revenue Collection process in New World ERP. On each business day, a receipt batch will automatically be created when the first payment is processed from the IVR. That payment and subsequent payments from that day will be collected in the batch as individual receipts.

Taking in Payments from the IVR

Collection Station

Tyler recommends creating a separate collection station for the IVR receipt batches, as it will help to clarify the source of the payments in reporting an inquiry.

Navigate to Logos Suite\ Revenue Collections\ Collection Stations in New World ERP. Create a new active collection station called IVR, with a dummy IP address.

Collection Station List
Collection Station - UB IVR (IVR)

Collection Station	Cash Drawer
Active <input checked="" type="checkbox"/>	Cash Drawer Name <input type="text"/>
Name <input type="text" value="UB IVR (IVR)"/>	Model <input type="text"/>
IP Address <input type="text" value="1.1.2.2"/>	Beginning Cash Balance
Print Receipts <input type="checkbox"/>	Effective Date <input type="text"/>
Printer	Start Date <input type="text"/>
Type <input type="text" value="Local Laser"/>	Amount <input type="text"/>
Name <input type="text"/>	
Receipt Format <input type="text"/>	
Default Validate Checks <input type="checkbox"/>	

Virtual Payment Source

The IVR will automatically create and populate batches of payments in Revenue Collections. Virtual Payment Source is where these unattended batches are defined and configured.

Navigate to Logos Suite\ Revenue Collections\ Virtual Payment Source in New World ERP. Click **New** to create a new Virtual Payment Source type, and then enter the appropriate values:

Virtual Payment Source List
Virtual Payment Source

Payment Source Type

Collection Batch Defaults	Receipt Management Settings
Department <input type="text" value="3510 - FINANCE"/>	Allow Users to Void Receipt <input type="checkbox"/>
Batch End Time <input type="text" value="8:00 PM"/>	Allow Users to Modify Receipt <input type="checkbox"/>
User <input type="text" value="City of Troy Treasu"/>	Allow Users to Add Manual Receipt <input type="checkbox"/>
Override Cash Account <input type="text"/>	
Collection Station <input type="text" value="Credit Card"/>	
Description <input type="text" value="IVR Payments"/>	

Module

Module	Payment

These values define defaults for the receipt batch that will be automatically created to hold IVR payments. Additionally, they define the business day for the system.

- Enter IVR in the *Payment Source Type***
- Department:** Enter the department responsible for reviewing and posting IVR payments in Revenue Collection.
- Batch End Time:** The close of business. Payments arriving after this time will be placed in a batch for the next business day. This timing may be dictated by settlement considerations on the credit card charges.
- User:** This will be the cashier who will be responsible for the auto generated receipt batch.
- Override Cash Account:** Used only when the standard cash account for payments should be overridden for IVR payments.
- Collection Station:** The collection station created in the previous step. We recommend using a collection station name that clearly indicates that the IVR system was the source of these payments.
- Description:** "UM IVR Payments". This description is just to clarify the purpose of this virtual payment source in the list of virtual payment sources.
- Allow checkboxes:** These determine the users' ability to maintain the receipts in the IVR batch. In order to keep IVR payments isolated from other payment, and to ensure that integrity and accountability is maintained, the recommended setting for these checkboxes is cleared.

Click **Save** to save the receipt batch settings.

Click the **New** button under Module, to create a new association to a payment code.

The screenshot shows a window titled "Payment Source Type". Inside, there is a "Module" field with the value "880E9B18-1672-4BF4-8126-55ADCBA40632" and a "Payment Code" dropdown menu set to "UT Payment - Utility Payment". Below these fields, there are two lists of transaction reference numbers. The "Available Transaction Reference Numbers: 13" list includes: License Type and Number (License Application), License Type and Number (License Bond), License Type and Renewal Number (License Charges), Licensee Number (Undesignated Balance), M/B Customer Number, M/B Invoice Type and Number, Permit Type and Number, Project Name and Activity, Special Assessments Invoice Number, Special Assessments Jurisdiction and Parcel Number, Special Assessments Payoff with Parcel Number and District, and UM Bad Debt Account Number and Batch. The "Selected Transaction Reference Numbers: 1" list contains "UM Account Number". There are four arrow buttons between the two lists for moving items. An "OK" button is at the bottom left.

A pop-up will appear. This in part defines the characteristics of the individual receipts that will be created and connects this setup with your instance of the IVR.

- Enter a GUID for module.** This must match the GUID in the **VendorConfig.XML** as described in “IVR Services Configuration” section above.
- Select a Payment Code.** This must be a payment code defined for use with the Utility Management Subledger.
- Select UM Account Number** from available transaction reference numbers in the multi-select box.

Note: The Finance Department may want to create a new payment code specifically for IVR payments, to allow for easier filtering of IVR receipts vs Non-IVR receipts in Revenue Collections inquiry and reporting.

Service Fee Configuration

When customers pay by credit card, service fees can be applied as either a flat amount or percentage of the payment amount. Multiple tiers can be defined with threshold amounts to charge different amounts or percentages depending on the transaction total. Depending on the software configuration, your organization can choose to either collect and record the fees using a separate payment code or the fees can be passed through and charged directly by the third party provider.

To create a service fee for credit card transactions, navigate to **Maintenance \Logos Suite \Revenue Collection \Service Fees** and follow the steps below:


1. Click **New**. The Service Fee Entry page will be displayed.

Service Fee List

Service Fee Entry

Active ☒


Pass-Through ☐

Start Date 06/04/2015 

Code

Description

Payment Code for Service Fee

Purchase Amount Up To	Service Fee Type	Service Fee Value
 Add new row		

Apply Service Fee to Payment Codes Selection

Available Payment Codes: 379

5.00% CSH fee - 5.00% CSH fee
Akron Rd ODOT - Akron Road ODOT Signals
B&P - Misc Sales - B&P - Misc Sales & Services
BZE - Bldg Permi - BZE - Building Permits
BZE - Ele Permit - BZE - Electric Permits
BZE-Comm Dev Pe - BZE - Commerical Devel Permits
Cable Misc Sales - Cable Misc Sales & Service
Cash Draw O/S - Cash Drawer Over/Short Util Off
CC Lease Pay - Community Center Lease Payments
CCA Costs - CCA Costs
CCA Receipts - CCA Receipts

Selected Payment Codes: 0

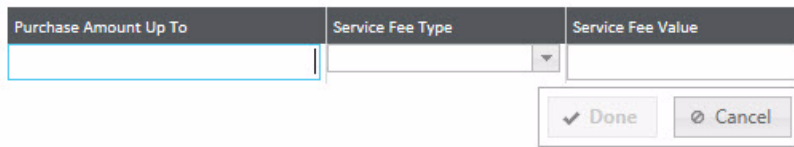
Save Save/New Reset Calculate Service Fee

2. If the service fee should be **Active**, select this check box.
3. Do not check the **Pass-Through** check box – this only used for integrated credit card payments.
4. Enter the **Start Date** for when the fee should start being applied. The default value is the current date for new entries. You may enter a different date or click the calendar icon to select one.
5. Enter a **Code** for the service fee. This is a required field that can accommodate up to 16 alphanumeric characters.
6. Enter a **Description** for the service fee. This is a required field that can accommodate up to 64 alphanumeric characters.
7. In the **Service Fee Type** drop-down, select from either **Flat Fee** or **Percentage**. This determines whether a set flat amount is charged as the service fee or if it should be a certain percentage of the payment amount.
8. The value entered in the **Service Fee Value** field will depend on the selection that was made in the previous drop-down. If you selected **Flat Fee**, enter a dollar amount with up to two decimal places. If **Percentage** was chosen, enter a percentage amount up to two decimal places. This field is required.

9. Select a **Payment Code for Service Fee** from the drop-down. A selection in this field is required. When a payment code is selected from the drop-down, the fee will be charged by your organization and recorded in Logos as a transaction. If using this method, the payment code that was set up in the section above should be selected.

Note: During the creation or modification of a service fee, the selected payment codes cannot be part of an open receipt batch or an error will be generated. Once the batches are posted, modifications including the deletion of the service fee can be made if necessary.

10. Click on Add new row in the grid to enable the fields where a single or multiple level of service fees can be set up.



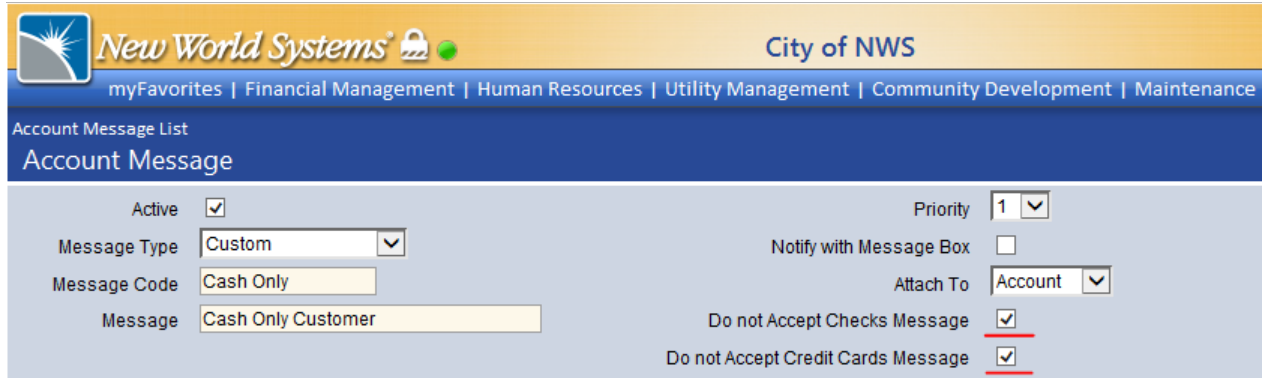
11. The Purchase Amount Up To field is not required if only a single flat fee or percentage is being set up for the service fee. A single row in the grid will charge the same service fee (either flat amount or percentage) for all transactions. If you are setting up a multitiered fee structure (e.g. more than one row), this field is required. Enter the maximum purchase amount for which the defined fee amount or percentage should be charged. The maximum amount that can be entered is 1,000,000.00.
12. In the Service Fee Type drop-down, select either Flat Fee or Percentage. A selection in this field is required.
13. In the Service Fee Value field, enter the amount that should be charged for the service fee. This field is required.
14. Once all required fields have been entered, click Done. The information will be added to the grid. If any edits need to be made to the entered information, click on the row to reactivate the entry fields.

Presenting Information to the Consumer

Restricted Payment Methods (Cash Only Flag) Configuration

Many IVR systems are capable of restricting specific customers from paying by check or credit card if they have a bad history with those payment types.

Accounts must be marked in New World ERP as **Do Not Accept Check Message** and/or **Do Not Accept Credit Card Message**. This is done by assigning an account message to the account and properly configuring the messages, so the system understands that there are payment tender restrictions.



New World Systems City of NWS

myFavorites | Financial Management | Human Resources | Utility Management | Community Development | Maintenance

Account Message List

Account Message

Active ☒

Message Type

Message Code

Message

Priority

Notify with Message Box ☐

Attach To

Do not Accept Checks Message ☒

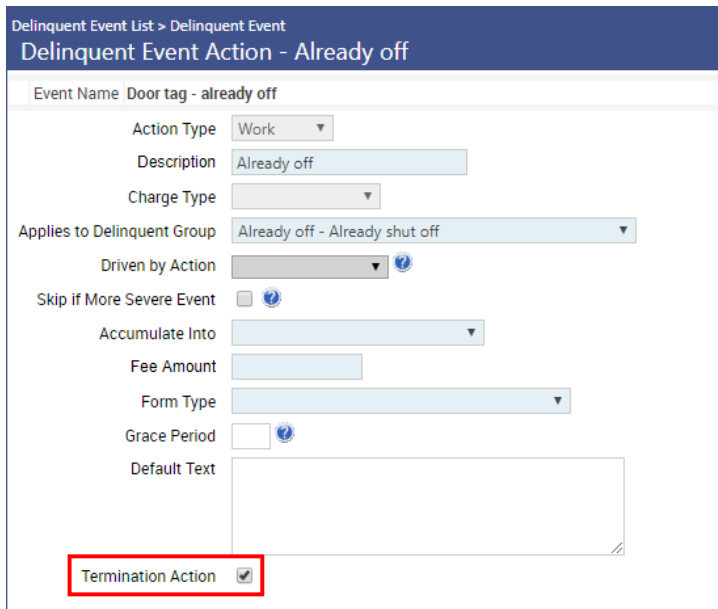
Do not Accept Credit Cards Message ☒

Note: At this point, you may want to replicate the setup steps for Collection Station, Payment Code, Virtual Payment Source, Service Fee, Shutoff Date, and Restricted Payment Method in both LIVE and TEST, so that this configuration will not have to be re-created when refreshing the LIVE New World ERP database over TEST.

Shut off Date Configuration

Many IVR systems can present a shut off date to delinquent consumers. In order to have this date available to the IVR, your shut off event needs to be tagged as such.

This step is optional. In order to enable the shut off date, you must have at least one delinquent event with at least one work type action flagged for termination action, i.e., a work type set up with a selected **Termination Action** check box (see image below). This is done via the Delinquent Event Action screen (**Maintenance > Utility Management > Delinquent Events**).



Delinquent Event List > Delinquent Event

Delinquent Event Action - Already off

Event Name

Action Type

Description

Charge Type

Applies to Delinquent Group

Driven by Action

Skip if More Severe Event ☐

Accumulate Into

Fee Amount

Form Type

Grace Period

Default Text

Termination Action ☒

WEB API TEST CLIENT

Customers have the option of testing the interface to ensure that it is installed properly, working as it should, and ready for the third party to test and code against.

To do this, Tyler provides a tool with the product for testing the installation and querying data from the Logos WebAPI. Developers can also use this tool to validate their work.

Due to the technical nature of the following information, this section is intended for either IT personnel or administrators with strong technical knowledge.

Initial Testing

Navigate to C:\Program Files (x86)\New World Systems\Logos Web API\ClientTool\Resources and open **TestConfig.xml** file in a text editor:

```
<config>
  <!--
    Http Status Codes
    200 - OK
    201 - Created
    400 - Bad Request
    401 - Unauthorized
    404 - Not Found
    500 - Internal Server Error
  -->
  <baseuri>http://localhost/Logos.WebApi/</baseuri>
  <authkey>bill</authkey>
  <authvalue>password</authvalue>
  <output>application/xml</output>
  <includeauthcred>True</includeauthcred>
  <includelogoscred>True</includelogoscred>
  <tests>
    <test active="True">
      <name>GetUtilityAccountInfo</name>
      <endpoint>api/UM/V1/Account/697320-001</endpoint>
      <httpmethod>GET</httpmethod>
      <logosid>697320-001</logosid>
      <logospin>2117</logospin>
      <expectedhttpcode>200</expectedhttpcode>
    </test>
  </tests>
</config>
```

This XML file has a number of tests defined to ensure all features are working correctly. Ensure the fields highlighted in yellow have valid values for your environment.

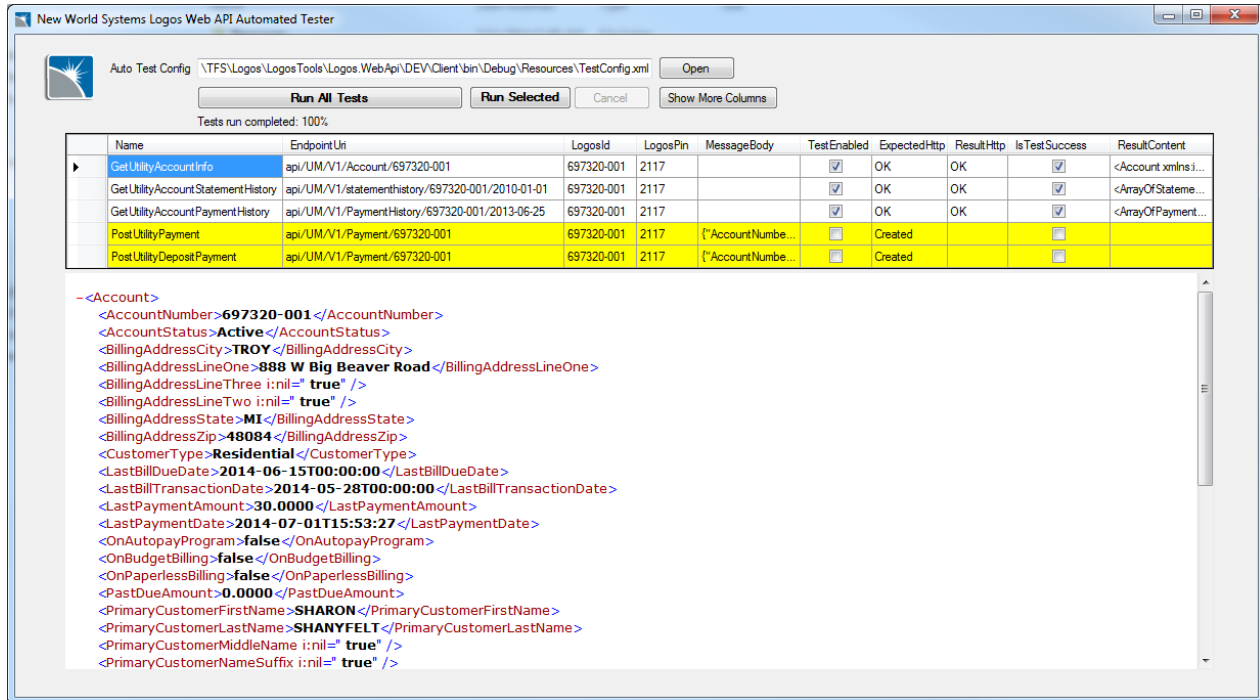
AuthKey is the vendor **username** as defined in the **VendorConfig.XML** file.

AuthValue is the vendor **password** as defined in the **VendorConfig.XML** file.

LogosID is the UM account number.

LogosPIN is the numeric part of the service address.

Launch the **Web API Test Client** by running C:\Program Files (x86)\New World Systems\Logos Web API\ClientTool\Logos.WebApi.Client.exe.



Click any column in the first row to select it, and click 'Run Selected' button to run the test. You should see an XML formatted result in the bottom pane, similar to the image above.

Additional Tests

The **PostUtilityPayment** and **PostUtilityDepositPayment** are disabled by default for security. Initiating these tests will post a utility payment to New World ERP, and will have significant impact on a LIVE installation.

In the **TestConfig.XML**, ensure that the highlighted fields below are valid for your data set before testing the payment functions. Change the date to a recent bill cycle to easily review the payment in UM Customer Service.

```
<test active="False">
  <name>PostUtilityPayment</name>
  <endpoint>api/UM/V1/Payment/697320-001</endpoint>
  <httpmethod>POST</httpmethod>
  <logosid>697320-001</logosid>
  <logospin>2117</logospin>
  <expectedhttpcode>201</expectedhttpcode>
  <body>
    <![CDATA[
      {"AccountNumber": "697320-001", "PayeeName": "John Travolta", "PaymentAmount": 30.0,
      "PaymentDate": "2014-06-25T16:44:36.4959176-05:00", "PaymentType": "Charge",
      "AuthorizationCode": "AuthCode", "TransactionCode": "TransCode", "Comment": "Payment via WebApi."}
    ]]>
  </body>
</test>
```

You can choose to run all tests at once or run them individually by clicking on a test row on grid and clicking the **Run Selected** button. If a test fails, the row will be highlighted in red. Disabled tests are highlighted in yellow. You can enable a test by checking the 'TestEnabled' checkbox or by changing the attribute named 'Active' to "true" in the XML file.

Similarly you can change all values from the test tool UI and run the tests with new values. Any manual changes you make in the test tool will not be saved to the TestConfig.XML file.

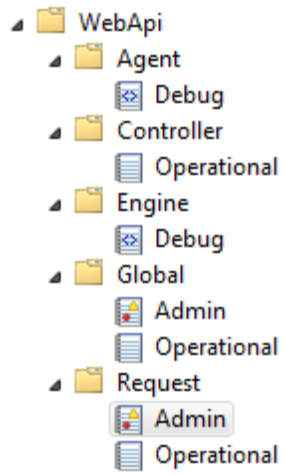
HANDOFF

If the processes in the Web API Test Client are successful, the service is functional, and ready for presentation to the third party IVR vendor. At a minimum, the vendor will need to know the **IP address** of the web server, and the vendor **username** and **password** from the **VendorConfig.XML** file. IVR vendors who have not developed against the Web API may need a copy of the *NWS Utility Management Integration Guide* for review.

TROUBLESHOOTING

The Logos Web API logs all requests in Windows Event Log. The logs can be viewed in **\Event Viewer \Applications** and **\Services \Logos \WebApi**.

Every incoming request is given a session id that can be used to track all event logs associated to that particular session. Administrators should monitor the **\Global \Admin** log for setup and application failures, and **\Request \Admin** log for failed login attempts.



New World ERP Utility Management Integration

Overview

This appendix describes the services that are available to integrate New World ERP Utility Management with non-related utility systems. The services provide discrete operations to allow an integrator to either access data or request an action to be performed by the system.

General Programming Guidance

The services provide operations that allow an external system to acquire operational data from New World ERP and, in some circumstances, to control the behavior of New World ERP. Services are published over the HTTP protocol using the Representational State Transfer (REST) pattern. System integrators can interact with the services using simple HTTP resource requests. This simple pattern supports a broad range of clients ranging from server-based systems to mobile devices.

In order to request a resource, the integrator must create an HTTP request directed to the URL specified for the desired resource. New World ERP will retrieve the resource at that location and return it to the requestor. The requestor is then free to use that resource without the expectation of any further interaction with New World ERP.

Security Considerations

Tyler New World will provide valid security credentials suitable for working with the approved operations. These credentials must be provided with each resource request. New World ERP will authenticate the credentials and authorize the caller for access for the requested operation on the given resource. The API uses HTTP basic authentication.

ALL resources requests must be made over a secure sockets layer (SSL) connection.

Creating the Resource Request

Generate the resource request by sending a supported HTTP verb (GET, DELETE, and POST) to a URL that specifies the location of the resource. The URL indicates how the request will be processed. For example, the resource at UM/v1/Account/137310-002 would return the account details for the account with the ID 137310-002. An example URL for this request might be <https://logos.net/um/v1/account/137310-002>.

The HTTP response contains any data returned by the resource request in the body of the response. The requestor may specify the data representation of the response as either XML or JSON.

Web API Client Sample

The following code sample demonstrates how to request an account from the service. This code implements a .Net client written with the C# programming language.

```
HttpClientHandler handler = new HttpClientHandler();

handler.Credentials = new NetworkCredential("3rdpartyusername", "3rdpartypassword");
handler.PreAuthenticate = true;

HttpClient client = new HttpClient(handler);

client.BaseAddress = new Uri("http://localhost/Logos.WebApi/");

var mediaType = new MediaTypeWithQualityHeaderValue("application/json");
client.DefaultRequestHeaders.Accept.Add(mediaType);

//logosid and logospin aren't always required.
client.DefaultRequestHeaders.Add("logosid", "137310-002");
client.DefaultRequestHeaders.Add("logospin", "1051");

HttpResponseMessage response = client.GetAsync("UM/v1/Account/137310-002").Result;

if (response.IsSuccessStatusCode)
{
    Console.Write(response.ToString());
    Console.WriteLine("");

    var readTask = response.Content.ReadAsStringAsync();
    readTask.Wait();

    Console.Write(readTask.Result);
}
```

Figure 1: C# implementation of a Web API client.

Response Body Formats

The response body returned from a resource request can be represented to the client as JSON or XML. JSON, or JavaScript Object Notation, is a lightweight data interchange format that can be read by both humans and computers. The data format consists of a number of name value pairs that represent the properties of an object. XML (Extensible Markup Language) is a more verbose data interchange format that consists of encoding rules that guide the use of markup and content to define an object.

application/json, text/json

```
{
  "AccountNumber": "00000001-001",
  "AccountStatus": "Active",
  "PrimaryCustomerLastName": "sample string 3",
  "PrimaryCustomerFirstName": "sample string 4",
  "PrimaryCustomerMiddleName": "sample string 5",
  "PrimaryCustomerNameSuffix": "sample string 6",
  "ServiceAddressHouseNumber": "sample string 7",
  "ServiceAddressAdditionalInfo": "sample string 8",
  "ServiceAddressStreetPrefix": "sample string 9",
  "ServiceAddressStreetType": "sample string 10",
  "ServiceAddressStreetPreType": "sample string 11",
  "ServiceAddressStreetName": "sample string 12",
  "ServiceAddressStreetSuffix": "sample string 13",
  "ServiceAddressApartment": "sample string 14",
  "TotalDueAmount": 15.0,
  "PastDueAmount": 16.0,
  "LastBillDueDate": "2013-11-12T12:40:10.1588387-05:00",
  "LastBillTransactionDate": "2013-11-12T12:40:10.1588387-05:00",
  "PrimaryPhoneNumber": "sample string 19",
  "PrimaryEmailAddress": "sample string 20",
  "LastPaymentAmount": 1.0,
  "LastPaymentDate": "2013-11-12T12:40:10.1608389-05:00",
  "OnAutopayProgram": true,
  "OnPaperlessBilling": true,
  "OnBudgetBilling": true,
  "CustomerType": "sample string 24",
  "RestrictedPaymentMethods": "sample string 25"
}
```

Figure 2: An example of data represented in the JSON format.

application/xml, text/xml

```
<Account xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://schemas.datacontract.org/2004/07/NewWorld.Logos.WebApi.Models.UM.Accounts">
  <AccountNumber>00000001-001</AccountNumber>
  <AccountStatus>Active</AccountStatus>
  <CustomerType>sample string 24</CustomerType>
  <LastBillDueDate>2013-11-12T12:40:10.1588387-05:00</LastBillDueDate>
  <LastBillTransactionDate>
    2013-11-12T12:40:10.1588387-05:00
  </LastBillTransactionDate>
  <LastPaymentAmount>1</LastPaymentAmount>
  <LastPaymentDate>2013-11-12T12:40:10.1608389-05:00</LastPaymentDate>
  <OnAutopayProgram>true</OnAutopayProgram>
  <OnBudgetBilling>true</OnBudgetBilling>
  <OnPaperlessBilling>true</OnPaperlessBilling>
  <PastDueAmount>16</PastDueAmount>
  <PrimaryCustomerFirstName>sample string 4</PrimaryCustomerFirstName>
  <PrimaryCustomerLastName>sample string 3</PrimaryCustomerLastName>
  <PrimaryCustomerMiddleName>sample string 5</PrimaryCustomerMiddleName>
  <PrimaryCustomerNameSuffix>sample string 6</PrimaryCustomerNameSuffix>
  <PrimaryEmailAddress>sample string 20</PrimaryEmailAddress>
  <PrimaryPhoneNumber>sample string 19</PrimaryPhoneNumber>
  <RestrictedPaymentMethods>sample string 25</RestrictedPaymentMethods>
  <ServiceAddressAdditionalInfo>sample string 8</ServiceAddressAdditionalInfo>
  <ServiceAddressApartment>sample string 14</ServiceAddressApartment>
  <ServiceAddressHouseNumber>sample string 7</ServiceAddressHouseNumber>
  <ServiceAddressStreetName>sample string 12</ServiceAddressStreetName>
  <ServiceAddressStreetPreType>sample string 11</ServiceAddressStreetPreType>
  <ServiceAddressStreetPrefix>sample string 9</ServiceAddressStreetPrefix>
  <ServiceAddressStreetSuffix>sample string 13</ServiceAddressStreetSuffix>
  <ServiceAddressStreetType>sample string 10</ServiceAddressStreetType>
  <TotalDueAmount>15</TotalDueAmount>
</Account>
```

Figure 3: An example of data represented as XML.

Error Messages

```
-<Error>
  <Message>Invalid vendor credential in http authorization header.</Message>
</Error>
```

Figure 4: An example of an error represented as XML.

```
{"Sid": "1", "Message": "Invalid vendor credential in http authorization header."}
```

Figure 5: An example of an error represented as JSON.

HTTP Status Codes

A request can be answered with any of the following:

RESULT	HTTP CODE	DESCRIPTION
SUCCESS	200	Request was processed successfully.
CREATED	201	Requested object created successfully.
BAD REQUEST	400	Input was not formatted properly.
UNAUTHORIZED	401	You are not allowed to access this resource.
NOT FOUND	404	Requested object was not found.
ERROR	500	Server was unable to process request.

Utility Management Resource Reference



Note: Data types with postfix '?' can have null values.

Account

The Account entity is used to return information about utility account requested by invoking following endpoint.

Endpoint	Request Header	Request Body	Response Header	Response Body
GET /{AccountNumber}	Authorization	N/A	HTTP 200	Account
	LogosId			
	LogosPIN			

Account Entity

Name	Data Type	Max Length	Note Type	Note Value
AccountBlocked	Boolean		Explain	Can be used to indicate the consumer should be blocked from automated account access such as web portals or IVR systems. True if "eSuite blocked reason" tied to account in Logos.
AccountBlockedReason	String?	128	Info	A text description of why the account is blocked. Values come from a validation set defined in Logos.
AccountNumber	String	24	Sample	00000001-001
			Format	Regex: ^([a-zA-Z0-9]*-[0-9]*)+\$
AccountStatus	String	10	Enum	Pending, Active, Inactive
BillingAddressCity	String?	32		
BillingAddressLineOne	String?	40		
BillingAddressLineThree	String?	40		
BillingAddressLineTwo	String?	40		
BillingAddressState	String?	32		
BillingAddressZip	String?	10		
CustomerType	String	32	Info	Values come from a validation set defined in Logos.
LastBillDueDate	DateTime?		Format	yyyy-MM-ddThh:mm:ss
LastBillTransactionDate	DateTime?		Format	yyyy-MM-ddThh:mm:ss
LastPaymentAmount	Decimal			
LastPaymentDate	DateTime?		Format	yyyy-MM-ddThh:mm:ss
OnAutopayProgram	Boolean		Explain	The customer has signed up for an automatic bill payment program offer by city.
OnBudgetBilling	Boolean		Explain	Also known as 'Leveled Billing'. Enable customers to avoid unpredictable bills by equalizing payments over a period of time.
OnPaperlessBilling	Boolean			
OnPaymentPlan	Boolean		Explain	The customer has a past due balance and has made arrangements with city to pay the past due in installments.
PastDueAmount	Decimal		Explain	Amount past due date
PaymentPlanRecentChargesAmountDue	Decimal		Info	Filled if OnPaymentPlan is true.
			Explain	Amount due for payment plan customers to be current on their ongoing charges.
PaymentPlanRecentChargesDueDate	DateTime?		Info	Filled if OnPaymentPlan is true.
			Explain	Due date for those ongoing charges.
PaymentPlanInstallmentAmountDue	Decimal		Explain	Filled if OnPaymentPlan is true.
			Info	Add PaymentPlanRecentChargesAmountDue to this amount to get AmountDue.
PaymentPlanInstallmentDueDate	DateTime?		Info	Filled if OnPaymentPlan is true.
			Explain	Next payment plan installment due date.
PrimaryCustomerFirstName	String?	20		
PrimaryCustomerLastName	String	50		
PrimaryCustomerMiddleName	String?	20		
PrimaryCustomerNameSuffix	String?	32	Info	Validation Set defined in Logos.
PrimaryEmailAddress	String?	50		
PrimaryPhoneNumber	String?	15	Sample	2603330000 (no dashes)
RestrictedPaymentMethods	String?	50	Enum	Check,CreditCard
			Explain	Check or CreditCard or Check,CreditCard

ServiceAddressAdditionalInfo	String?	32		
ServiceAddressApartment	String?	32		
ServiceAddressCity	String	32		
ServiceAddressHouseNumber	String	40		
ServiceAddressState	String	32		
ServiceAddressStreetName	String	50		
ServiceAddressStreetPrefix	String?	3		
ServiceAddressStreetPreType	String?	10		
ServiceAddressStreetSuffix	String?	3		
ServiceAddressZip	String	5		
ServiceAddressZip4	String	4		
ShutOffDate	DateTime?		Caution	Setup required on Logos to enable this.
			Explain	Filled for account with a past due amount.
			Caution	Not filled if account on payment plan.
TotalDueAmount	Decimal		Explain	Amount due as of today
			Caution	Don't use this if account is on budget bill or payment plan.

Usage

Use the HTTP GET verb to request this resource in the form UM/v1/Account/{accountNumber}. New World ERP will provide the account details for a valid account number in the HTTP response body.

Payment

Posts a payment to the account with the given account number.

Endpoint	Request Header	Request Body	Response Header	Response Body
POST/{AccountNumber}	Authorization	Payment	HTTP 201	Payment
	LogosId			
	LogosPIN			

Payment Entity

Name	Data Type	Max Length	Note Type	Note Value
AccountNumber	String	24	Sample	00000001-001
			Format	Regex: ^([a-zA-Z0-9]*-[0-9]*)+\$
AuthorizationCode	String?	32	Explain	Payment transaction unique identifier
				Credit card authorization code
Comment	String?	256	Explain	Appears on receipt in comments section
PayeeName	String?	64		
PaymentAmount	Decimal			
PaymentDate	DateTime			
PaymentType	String	10	Enum	Cash,Check,Charge
FeeAmount	Decimal?		Explain	Service fee, transaction fee or convenience fee.
			Caution	Setup required in Logos.
FeeType	String?	15	Enum	Real,PassThrough
			Explain	Real: fee collected by city
				PassThrough: fee collected by gateway
FeeAuthorizationCode	String?	32		
TransactionCode	String?	128	Explain	3 rd party transaction reference code
				To tie the resulting receipt in Logos with a transaction in external system

Usage

Use an HTTP POST verb to send data to this resource. The complete payment entity must be specified in the body of the message.

Payment History

Returns details of payment history for account with the provided account number.

Endpoint	Request Header	Request Body	Response Header	Response Body
GET /{AccountNumber}/StartDate	Authorization	N/A	HTTP 200	PaymentHistory
	LogosId			
	LogosPIN			

Payment History Entity

Name	Data Type	Max Length	Note Type	Note Value
AccountNumber	String	24	Sample	00000001-001
			Format	Regex: ^([a-zA-Z0-9]*-[0-9]*)+\$
BillingDate	DateTime?		Explain	Billing date of bill paid by the payment. It may be null in certain situations e.g. over payment.
PaymentAmount	Decimal			
PaymentDate	DateTime			
ServiceFromDate	DateTime?		Explain	From date of bill paid by the payment. It may be null in certain situations e.g. over payment.
ServiceToDate	DateTime?		Explain	To date of bill paid by the payment. It may be null in certain situations e.g. over payment.
TransactionDescription	String	50	Enum	Adjustment to Correct a Payment
				Adjustment to Correct a Bill
				Refund
				Balance Moved to Bad Debt Collections
TransactionType	String	10	Enum	Payment,Adjustment

Usage

Use the HTTP GET verb to request this resource in the form UM/v1/PaymentHistory/{accountNumber}/{startDate}. New World ERP will provide the payment history details for a valid account number in the HTTP response body. Start date format must be yyyy-MM-dd.

Statement History

Returns details for statement history for account with the provided account number.

Endpoint	Request Header	Request Body	Response Header	Response Body
GET /{AccountNumber}/StartDate	Authorization	N/A	HTTP 200	StatementHistory
	LogosId			
	LogosPIN			

Statement History Entity

Name	Data Type	Max Length	Note Type	Note Value
AccountNumber	String	24	Sample	00000001-001
			Format	Regex: ^([a-zA-Z0-9]*-[0-9]*)+\$
Adjustments	Decimal		Explain	Adjustment amount included in statement. Adjustment includes activities like bill/payment correction, deposit refund etc.
BalanceAtBilling	Decimal		Explain	Balance due prior to this statement
BalanceDue	Decimal			
BalanceDueAfterDueDate	Decimal		Explain	Balance due after PenaltyDate
BillDate	DateTime		Explain	Statement date
BillFromDate	DateTime?		Explain	Statement start date
BillToDate	DateTime?		Explain	Statement end date
CurrentCharges	Decimal			
CurrentMeterReadDate	DateTime?			
DueDate	DateTime		Explain	Due date as shown on bill
LastPaymentDate	DateTime?			
Payments	Decimal		Explain	Payment amount included in statement
Penalties	Decimal		Explain	Penalty amount included in statement
PenaltyDate	DateTime?		Explain	Date penalties will be applied to unpaid bill depending on system settings
				May be the same as due date or may represent a grace period after due date
			Usage	If null then use DueDate
PreviousBalance	Decimal		Explain	Balance shown on previous statement
PreviousMeterReadDate	DateTime?			

Usage

Use the HTTP GET verb to request this resource in the form UM/v1/StatementHistory/{accountNumber}/{startDate}. New World ERP will provide the statement history details for a valid account number in the HTTP response body. Start date format must be yyyy-MM-dd.

Consumption History

Returns details for consumption history for account with the provided account number.

Endpoint	Request Header	Request Body	Response Header	Response Body
GET /{AccountNumber}/StartDate	Authorization	N/A	HTTP 200	ConsumptionHistory
	LogosId			
	LogosPIN			

Consumption History Entity

Returns details for consumption history for account with the provided account number.

Name	Data Type	Max Length	Note Type	Note Value
AccountNumber	String	24	Sample	00000001-001
			Format	Regex: ^([a-zA-Z0-9]*-[0-9]*)+\$
BillingDate	DateTime			
Services	List<UtilityService>			

Utility Service Entity

Name	Data Type	Max Length	Note Type	Note Value
ServiceCode	String	16	Sample	ELECTRIC
ServiceDescription	String	32	Sample	Electric Service
Meters	List<ServiceMeter>			

Service Meter Entity

Name	Data Type	Max Length	Note Type	Note Value
CurrentMeterReadDate	DateTime?			
DaysBetweenReads	Integer			
MeterNumber	String	20	Sample	700033, G074560
PreviousMeterReadDate	DateTime?			
Measurements	List<MeterMeasurement>			

Meter Measurement Entity

Name	Data Type	Max Length	Note Type	Note Value
AverageDailyUsage	Decimal			
BilledConsumption	Decimal			
ConsumptionUnits	String	100	Sample	KWH
CurrentMeterRead	Decimal			
MeasurementType	String	128	Sample	Electric Reading
PreviousMeterRead	Decimal			

Usage

Use the HTTP GET verb to request this resource in the form UM/v1/ConsumptionHistory/{accountNumber}/{startDate}. New World ERP will provide consumption history details for a valid account number in the HTTP response body. Start date format must be yyyy-MM-dd.

Getting Started

Online Resources

The following online resources provide general programming knowledge for integrators wishing to use the .Net framework to consume REST APIs.

1. Use of HTTP Codes <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>
2. What is Web API <http://www.asp.net/web-api>
3. Videos on Web API <http://www.asp.net/web-api/videos>
4. How to create a Web API Client <http://www.asp.net/web-api/overview/web-api-clients/calling-a-web-api-from-a-net-client>

Logos Web API Test Client

A tool is provided with the product to query data from Logos Web API. Developers can use this tool to validate their work.

Querying Web API

1. Extract provided zip file and navigate to Resources folder and open TestConfig.xml file in a text editor.

```
<config>
  <!--
    Http Status Codes
    200 - OK
    201 - Created
    400 - Bad Request
    401 - Unauthorized
    404 - Not Found
    500 - Internal Server Error
  -->
  <baseuri>http://localhost/Logos.WebApi/</baseuri>
  <authkey>bill</authkey>
  <authvalue>password</authvalue>
  <output>application/xml</output>
  <includeauthcred>True</includeauthcred>
  <includelogoscred>True</includelogoscred>
  <tests>
    <test active="True">
      <name>GetUtilityAccountInfo</name>
      <endpoint>api/UM/V1/Account/697320-001</endpoint>
      <httpmethod>GET</httpmethod>
      <logosid>697320-001</logosid>
      <logospin>2117</logospin>
      <expectedhttpcode>200</expectedhttpcode>
    </test>
  </tests>
</config>
```

Logos Web API Test Client Configuration

This XML file has a number of tests defined to help query all features. Ensure fields highlighted in yellow have valid values for your environment. 'AuthKey' is vendor username and 'AuthValue' is vendor password (Ask city for the credentials). Logos Id is the account number and Logos PIN is the number part of service address as shown in the following image:



Account Information

Location

Account Number **106790-002**

Service Address **627 DOUGLAS ST**
Troy MI 48084-228

Carrier Route

Delivery Point

Billing Profile **First Series**

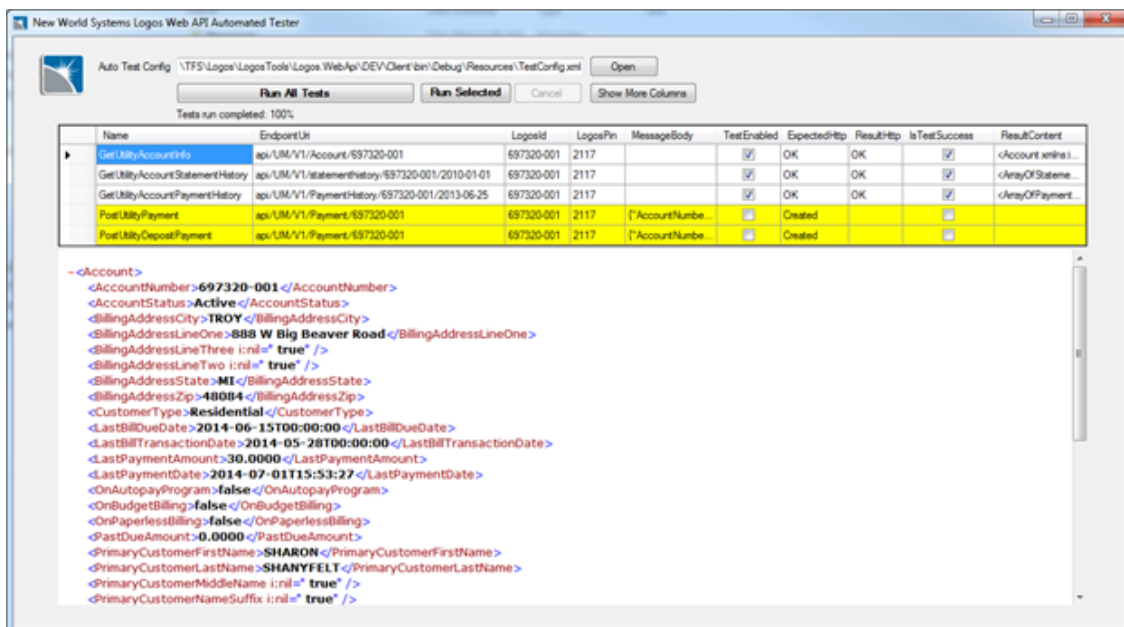
Customer

Name **ABRAHAM R ADAMS**

Phone **(10) 084-5248**

Logos Web API Customer Service Screen.

2. Launch Web API Test Client by running Logos.WebApi.Client.exe.



New World Systems Logos Web API Automated Tester

Auto Test Config: \TFS\Logos\Tools\Logos.WebApi\DEV\Client\bin\Debug\Resources\TestConfig.xml [Open]

[Run All Tests] [Run Selected] [Cancel] [Show More Columns]

Tests run completed: 100%

Name	EndpointUrl	LogosId	LogosPin	MessageBody	TestEnabled	ExpectedHttp	ResultHttp	IsTestSuccess	ResultContent
GetUtilityAccountInfo	api/UM/V1/Account/697320-001	697320-001	2117		<input checked="" type="checkbox"/>	OK	OK	<input checked="" type="checkbox"/>	<Account xmlns=...
GetUtilityAccountStatementHistory	api/UM/V1/StatementHistory/697320-001/2010-01-01	697320-001	2117		<input checked="" type="checkbox"/>	OK	OK	<input checked="" type="checkbox"/>	<ArrayOfStateme...
GetUtilityAccountPaymentHistory	api/UM/V1/PaymentHistory/697320-001/2013-06-25	697320-001	2117		<input checked="" type="checkbox"/>	OK	OK	<input checked="" type="checkbox"/>	<ArrayOfPayment...
PostUtilityPayment	api/UM/V1/Payment/697320-001	697320-001	2117	[{"AccountNumber...	<input checked="" type="checkbox"/>	Created		<input checked="" type="checkbox"/>	
PostUtilityDepositPayment	api/UM/V1/Payment/697320-001	697320-001	2117	[{"AccountNumber...	<input checked="" type="checkbox"/>	Created		<input checked="" type="checkbox"/>	

```

<Account>
  <AccountNumber>697320-001</AccountNumber>
  <AccountStatus>Active</AccountStatus>
  <BillingAddressCity>TROY</BillingAddressCity>
  <BillingAddressLineOne>888 W Big Beaver Road</BillingAddressLineOne>
  <BillingAddressLineThree i:nil="true" />
  <BillingAddressLineTwo i:nil="true" />
  <BillingAddressState>MI</BillingAddressState>
  <BillingAddressZip>48084</BillingAddressZip>
  <CustomerType>Residential</CustomerType>
  <LastBillDueDate>2014-06-15T00:00:00</LastBillDueDate>
  <LastBillTransactionDate>2014-05-28T00:00:00</LastBillTransactionDate>
  <LastPaymentAmount>30.0000</LastPaymentAmount>
  <LastPaymentDate>2014-07-01T15:53:27</LastPaymentDate>
  <OnAutopayProgram>false</OnAutopayProgram>
  <OnBudgetBilling>false</OnBudgetBilling>
  <OnPaperlessBilling>false</OnPaperlessBilling>
  <PastDueAmount>0.0000</PastDueAmount>
  <PrimaryCustomerFirstName>SHARON</PrimaryCustomerFirstName>
  <PrimaryCustomerLastName>SHANYFELT</PrimaryCustomerLastName>
  <PrimaryCustomerMiddleName i:nil="true" />
  <PrimaryCustomerNameSuffix i:nil="true" />

```

Logos Web API Test Client Screen

- Click any column in first row and then click the **Run Selected** button to run the test. You should see results similar to the image above.

There are tests that make changes to the system and, for security purposes, are disabled by default. Ensure that the fields highlighted in yellow below have valid values for your environment. The following test will post a utility payment to New World ERP.

```
<test active="False">
  <name>PostUtilityPayment</name>
  <endpoint>api/UM/V1/Payment/697320-001</endpoint>
  <httpmethod>POST</httpmethod>
  <logosid>697320-001</logosid>
  <logospin>2117</logospin>
  <expectedhttpcode>201</expectedhttpcode>
  <body>
    <![CDATA[
      {"AccountNumber": "697320-001", "PayeeName": "John Travanni", "PaymentAmount": 30.0,
        "PaymentDate": "2014-06-25T16:44:36.4959176-05:00", "PaymentType": "Charge",
        "AuthorizationCode": "AuthCode", "TransactionCode": "TransCode", "Comment": "Payment via WebApi."}
    ]]>
  </body>
</test>
```

Logos Web API Test Client Example of POST.

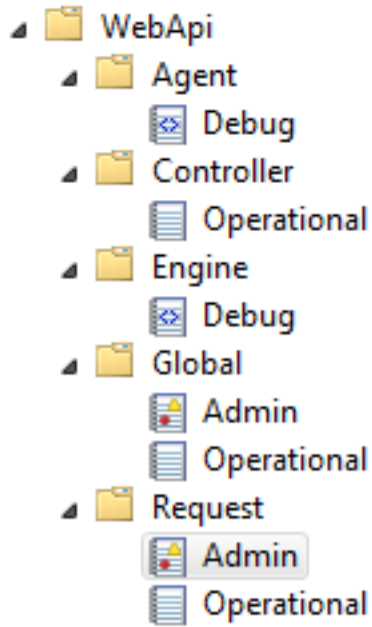
You can choose to run all tests at once or run them individually by clicking on test row on grid and clicking **Run Selected** button.

If a test fails, the row will be highlighted in red. Disabled tests are highlighted in yellow. You can enable a test by checking the **TestEnabled** check box or by changing the attribute named active to true in XML file.

Similarly you can change all values from the test tool UI and run the tests with new values. The changes will be lost when you exit the test tool. Results of the test run can be viewed in the area below the test grid by clicking on a test row within the grid.

Event Logging

Logos Web API logs all requests in Windows Event Log with information for administrators and developers. The logs can view in Event Viewer > Applications and Services Logs > Logos > WebApi.



Logos WebAPI Logs in Event Viewer

Every incoming request is given a session ID that can be used to track all event logs associated to that particular session. You may need to request access to the Web API machine in order to access Event Viewer logs.