



PL22063

Fusion Lifecycle's Evented Web Is Off the Hook

Michael Pares
Greenpoint Technologies

Martin Gasevski
Autodesk, Inc.

Doug McLean
Jitterbit

Learning Objectives

- Learn how to move past custom APIs and middleware integration approaches
- Address costly product customizations
- Learn about avoiding unique specialized product integrations
- Learn how to eliminate complex point-to-point connections

Description

This class will explain Fusion Lifecycle's (formerly Autodesk PLM 360) native support for connecting cloud, on-premise, social, and mobile technology without changing its existing architecture and site configurations—all accomplished with very little administrative effort. We will demonstrate how a product lifecycle management (PLM) site administrator can configure the built-in Evented Web features, and extend such needs programmatically to effortlessly and in real-time interconnect data, people, and processes with numerous other business solutions.



Your AU Expert(s)

Michael Pares

Senior Manager, Software Development, Greenpoint Technologies

Michael Pares manages software development, integration, and improvement at Greenpoint Technologies, a Boeing Business Jet (BBJ) Completion Center that provides turnkey interior completions for private individuals and heads of state. He has spent the last 10 years customizing and implementing a wide variety of enterprise applications for Greenpoint and its subsidiaries, and he is passionate about software and the creation of customer value through agile project management and rapid response to change. Pares and his team implemented Autodesk PLM 360 software (now Fusion Lifecycle software) in August of 2013, and they continue to enhance and add value through monthly or bi-weekly feature releases.

Martin Gasevski

Product Manager, Fusion Lifecycle, Autodesk, Inc.

Martin Gasevski is a product manager of the Fusion Lifecycle (formerly PLM 360) product, operating from Autodesk, Inc.'s, Michigan office. He has been actively involved in the product lifecycle management (PLM) and product data management (PDM) space for about 15 years as a software developer, success engineer, and product manager. In his present role, Gasevski contributes to the product roadmap at strategic and tactical levels, driving market and business requirements to engineering teams. His responsibilities are core PLM functionalities; product extensibility strategy ranging from REST API, scripting and eventing engines, and enterprise integrations; and technological partnership solutions and innovations.

Doug McLean

Alliances Director, Jitterbit

Doug manages Jitterbit ISV Alliance partnerships including Autodesk and related Consulting partners, with over 20 plus years of channel and business development experience. Doug was a founder of CloudTrigger, a Salesforce.com consultancy which grew to one of the top 25 Implementation partners for Salesforce.com prior to an acquisition. Doug resides in the Austin Texas area, in Wimberly Texas. Where he and his wife provide Equine Therapy services to local special needs kids and adults with their herd of miniature horses and donkeys.



Introduction

In Fusion Lifecycle you go to the Integration menu and choose Slack. There is a textbox for a New Customer webhook. You drop in your Slack token. Now when a new customer is created in Fusion Lifecycle, the Evented Web will send a message to your Slack channel with details about the customer. There is no simpler way to implement integration to all of the apps that run your business.

In the past, there was just Fusion Lifecycle

One could not simply point and click and integrate systems. Intimate knowledge of javascript, web apis, and security was necessary to implement even the most basic integration scenario, such as sending a message to a Slack channel.

Scripting

It is possible of course to do just about anything using scripting. It is not easy or cheap when compared with Evented Web capabilities. The general pattern would be to create a library function that handles the technical details of sending the message to the other system. Then import the library to an action script and call the function passing the message to the function. Then tie the action script to whatever is appropriate such as the create item event, a workflow transition, or an on-demand script.

Scripting Example

Here is an example of sending a message to Slack via script written directly in Fusion Lifecycle. I created a library function called `sendMessageToSlack` that accepts some message text as a parameter. I can call this library script from any action script where I import it. This is as easy as it gets for scripting, and this is a very simplistic example. Sending a message to another system could be much more complicated.

The screenshot displays the Fusion Lifecycle interface with two main sections: 'Action' and 'Library'.

Action Section:

- Action:** AU2016_Slack_Integration_Test
- Where Used:** (empty)
- Testing Slack message posting** (text input)
- Libraries:** AU2016_Slack_Integration, GTI_UserLib, GTI_GlobalVars
- Script:**

```
1 var transitionName = item.workflowActions[0].transition.shortName;
2 var transitionComment = item.workflowActions[0].comments;
3 var transitionUser = getUsername(item.workflowActions[0].userID);
4 var slackMessage = transitionName + ' has been performed by ' + transitionUser + ' on ' + item.descriptor.descriptor + '\n\n' + transitionComment;
5 sendMessageToSlack(slackMessage);
6
```

Library Section:

- Library:** AU2016_Slack_Integration
- Where Used:** (empty)
- Function to post json message request in a slack channel** (text input)
- Script:**

```
1 function sendMessageToSlack(messageText){
2   var SlackURL = 'https://hooks.slack.com/services/';
3
4   var xhr = new XMLHttpRequest();
5   xhr.open('POST', SlackURL, true);
6   xhr.setRequestHeader('Content-Type', 'application/json');
7   xhr.send(JSON.stringify({"text": messageText}));
8 }
```



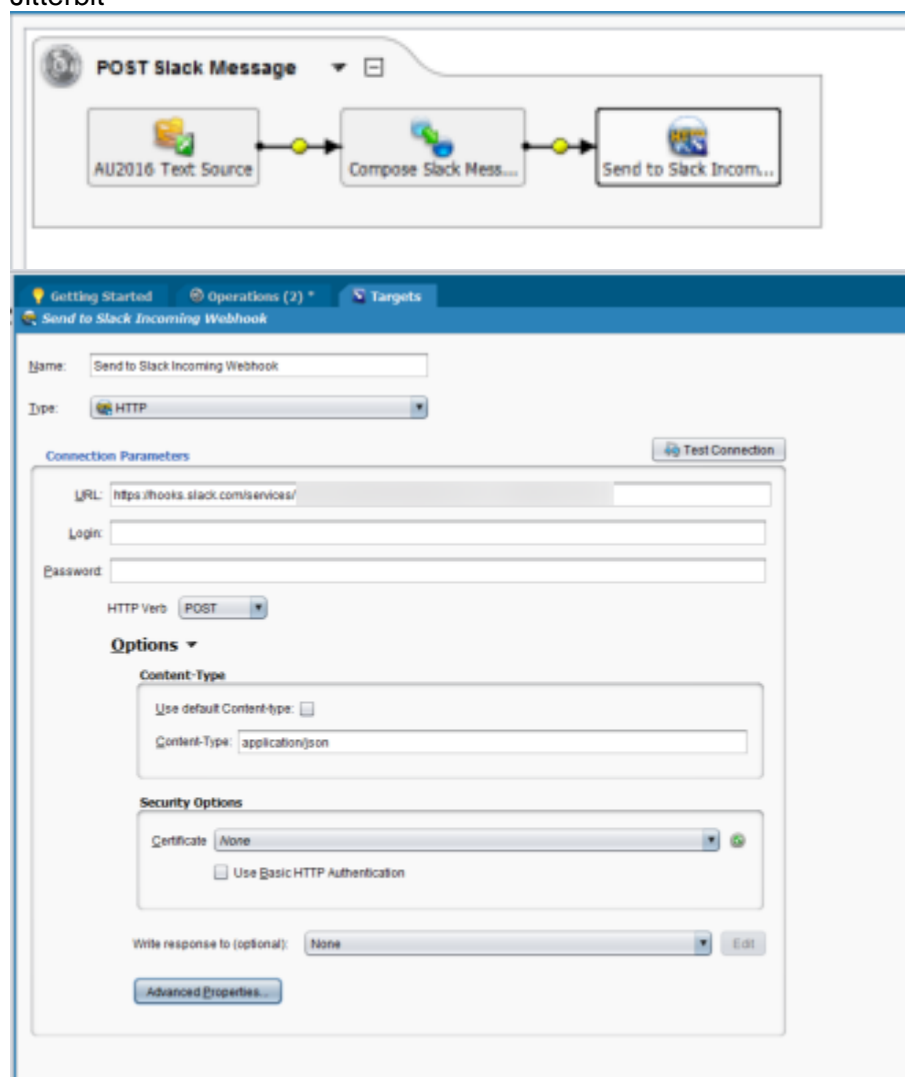
One thing to consider in this approach is timeouts! If your script is too big, too complex, it may run too long and your integration may not work. That may force you to use another method to achieve a complex integration, which is what we will talk about next. Anything you do with scripting and API needs to be asynchronous. Don't send large amounts of data this way.

Middleware - Jitterbit Orchestrations

For more complex integrations we need middleware like Jitterbit to handle the heavy lifting. This makes integration worlds easier but may still require people on staff with training and experience or assistance from consultants.

Jitterbit Example

Using middleware we can abstract the work required to send messages to many different types of systems. This is good because you can simplify the message being sent from Fusion Lifecycle. Here is an example of sending a message to Slack using Jitterbit





Jitterbit Harmony API

To tie it all together you need to have a location to send a message to. [Setting up a Jitterbit Harmony API](#) is a very simple way to accomplish this.

Harmony Example

Give your API a name which is added at the end of the URL that you will call. You then specify which Jitterbit project and orchestration will be triggered when your API URL receives an incoming message. You can then call upon jitterbit's [predefined global variables](#) to grab parameters or anything else in the payload of the message received and use it in your transformation. For example, in this API I am triggering the Slack Orchestration from the previous example and sending the message to slack whenever a message is received by the Harmony API

Edit Jitterbit Custom API Services
Base URL: GreenpointTechnologies.jitterbit.net
Service URL: http://GreenpointTechnologies.jitterbit.net/GreenpointDev/1.0/AU2016.html

Summary

Environment: Description:

Name:

Service Root:
(i.e. Public API Name)

Version:

Assigned Operations

Method	Project	Operations to Trigger on Request	Response	Action
ALL	AU2016	MAIN Incoming Webhook	system variable	Edit Delete

[Assign New Jitterbit Operations](#)

Authentication

Select Profile: [Assign](#)

Profile Name	User Name	Action
No Authentication Profiles Assigned, Anonymous Access Enabled		

Settings

SSL Only: ☐ Enable CORS: ☐ Debug Until:

Timeout: Seconds

[Update](#) [Cancel](#)

Assigned Operations

Method	Project	Operations to Trigger on Request	Response	Action
ALL	AU2016	MAIN Incoming Webhook	system variable	Edit Delete

Method Type:

Project:

Operation:

Response: ☐ final target ☒ system variable ☐ no response

[Update](#) [Cancel](#)

[Assign New Jitterbit Operations](#)

Learning Objective 1

[Learn how to move past custom APIs and middleware integration approaches]

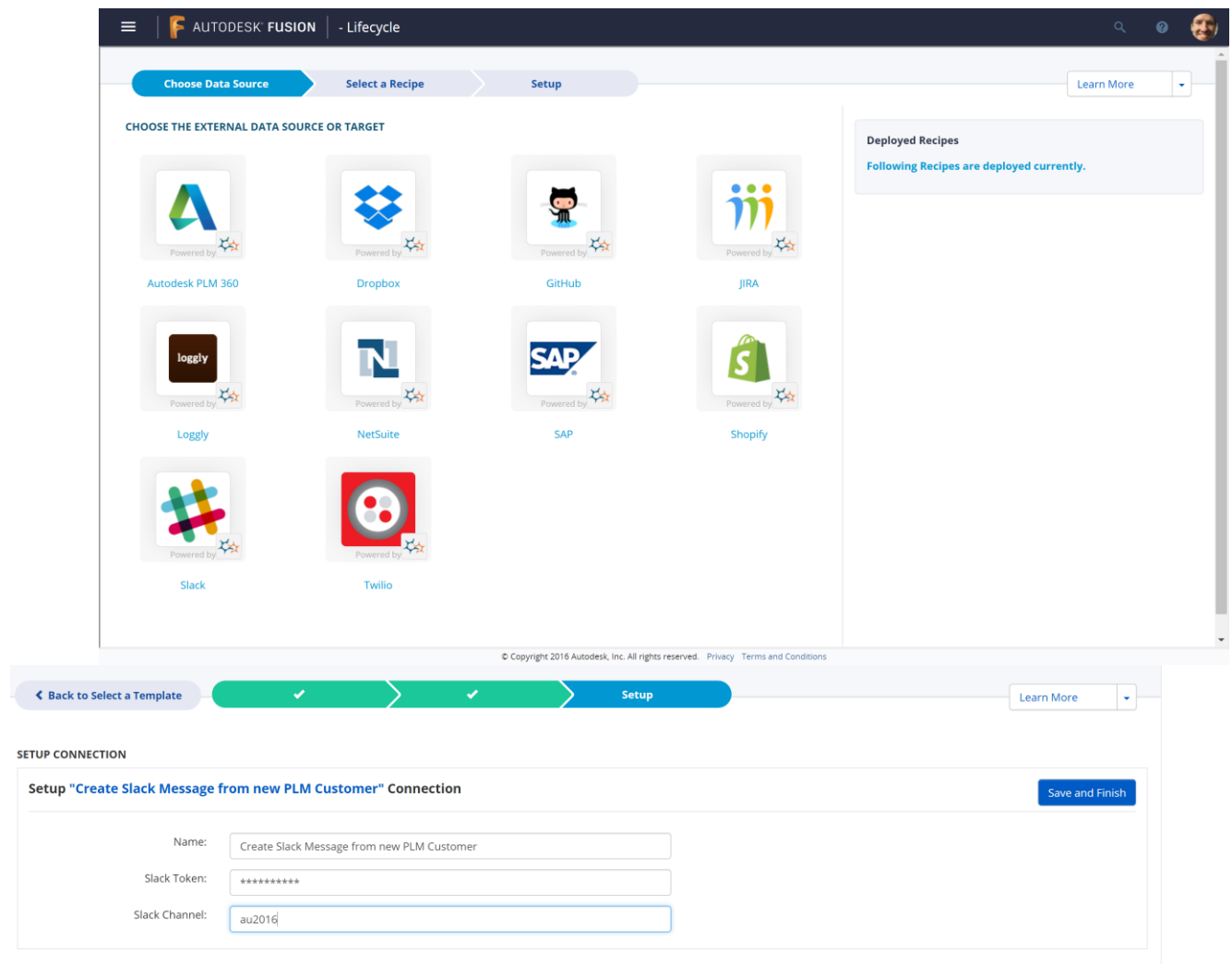
The examples above are pretty cool but they are also pretty simple. In truth middleware integrations can range from simple to very complex and scripting approaches can border on impossible. It would be really nice to have an integration experience that didn't take days and weeks to develop, and didn't require a developer or expert knowledge of web communication methods.

In Steps Evented Web

Here to grant those wishes is the Evented Web experience in Fusion Lifecycle, which allows you click and pick integration to a growing inventory of popular cloud-based applications that can easily integrate to your Fusion Lifecycle tenant.

Send a message to Slack using Citizen Integrator

In this example, you can see that sending a message to slack when there is a new customer is as easy as pasting in your slack token and channel and then save and finish.



The screenshot displays the Autodesk Fusion Lifecycle interface for setting up a new integration. The top navigation bar shows the Autodesk Fusion logo and the 'Lifecycle' tab. The main content area is titled 'CHOOSE THE EXTERNAL DATA SOURCE OR TARGET' and features a grid of application icons, each labeled 'Powered by' and the application name: Autodesk PLM 360, Dropbox, GitHub, JIRA, Loggly, NetSuite, SAP, Shopify, Slack, and Twilio. A 'Deployed Recipes' sidebar on the right indicates that no recipes are currently deployed. Below the grid, a progress bar shows the steps: 'Back to Select a Template' (with a left arrow), 'Select a Recipe' (with a checkmark), and 'Setup' (with a right arrow). The 'Setup' step is active, leading to the 'SETUP CONNECTION' section. This section is titled 'Setup "Create Slack Message from new PLM Customer" Connection' and includes a 'Save and Finish' button. The setup form contains three fields: 'Name' (pre-filled with 'Create Slack Message from new PLM Customer'), 'Slack Token' (masked with asterisks), and 'Slack Channel' (pre-filled with 'au2016').



Learning Objective 2

[Address costly product customizations]

Now that you have seen how simple it is to use Citizen integrator to connect systems with point and click you can take off on your own and prototype solutions for your users in record time. However because this is beta software, not all the bells and whistles are available yet. This means that your choices today for what workspaces and what applications are connected are actually rather limited. But don't despair because...

Here is what I have learned about creating webhooks in Evented Web

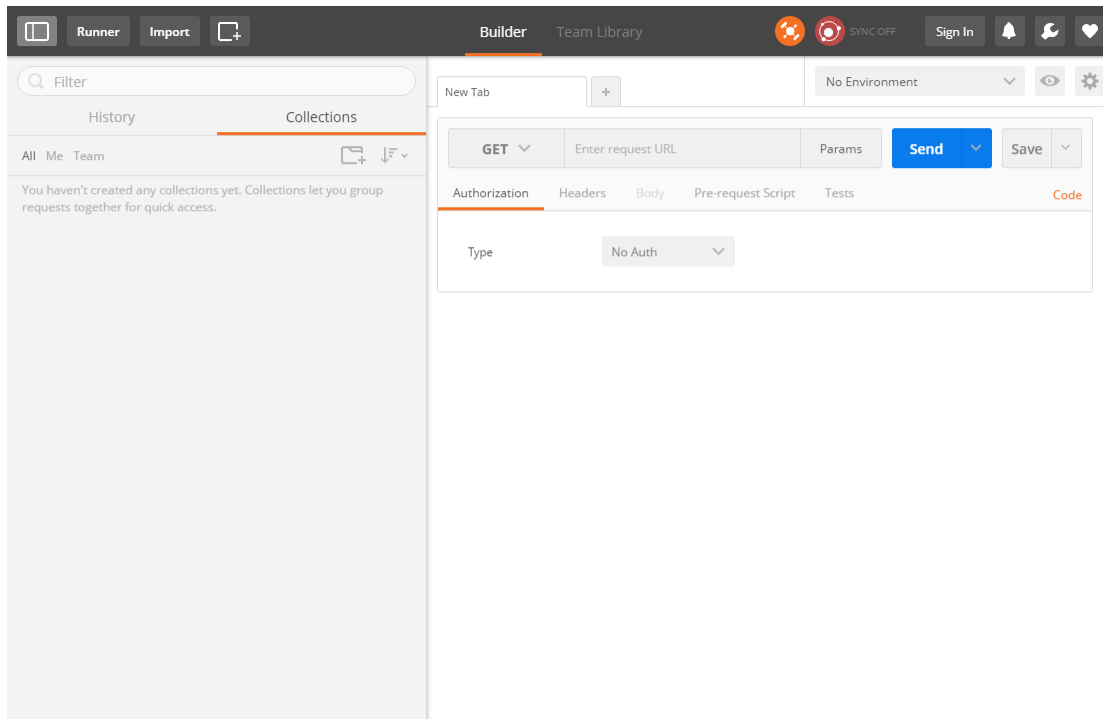
You can still create integrations that remove the need for Fusion Lifecycle scripting (and avoid timeouts). You just have to do it behind the scenes for now. Here is your step by step guide that decodes the [beta documentation](#) and that can get you creating Evented Web integrations with minimal effort.

First, you must create an application on Forge

You do this so that you can get access to a Client ID and Client Secret that will be created for you when you create your application. You need those for authentication (OAuth) when you do the next step, obtaining an access token. That's it. You don't need the App on Forge for any other purpose at this point in time. The App Name is anything you want, Description anything you want, Callback URL is the location you want your token to be returned to when you request it, which in our case is Postman, and Website URL is anything you want.

Then you obtain an access token using Postman

Postman is so cool. No, seriously. It took me a little bit to figure it out but before you go any further in this document, go get the [postman chrome app](#) and check out their documentation. You can use the free version for everything in this handout but you will be tempted to buy it after you see how it can organize all of your API requests and create documentation for you and write your code for you. Make sure you use the orange rocketman app rather than the classic postman app. If you are used to using the Advanced REST client for your Fusion Lifecycle API testing you may quickly replace it with Postman and the Postman Interceptor which syncs your chrome browser cookies. Anyways get Postman and then you will be greeted with this.



I tried to follow the instructions for getting a [3-Legged Token on the Forge website](#) and I did not know what to do from this point so here is the cheat code.

1. Select OAuth 2.0 under Type
2. Get New Access Token
3. Name the Token anything you want, such as JBAccessToken
4. The Auth URL is
<https://developer.api.autodesk.com/authentication/v1/authorize>
5. The Access Token URL is
<https://developer.api.autodesk.com/authentication/v1/gettoken>
6. The Client ID and secret are the same ones from your Forge app!
7. Set the Grant Type to Authorization Code
8. Make sure Request access token locally is unchecked
9. Request Token



GET NEW ACCESS TOKEN

Request a new access token to add it to your list of tokens
On clicking Request Token, you will be redirected to the Auth URL where you can enter the user's credentials and request for a token

Callback URL

https://www.getpostman.com/oauth2/callback
Set this as the callback URL in your app settings page.

Token Name

JBAccessToken

Auth URL

https://developer.api.autodesk.com/authenticat

Access Token URL

https://developer.api.autodesk.com/authenticat

Client ID

Client Secret

Scope (Optional)

Grant Type

Authorization Code
☐ Request access token locally

Cancel

Request Token

Then you will enter your Autodesk credentials and receive your token
When you Request Token then you are asked for your Autodesk account name and password. Enter them to enable Postman to obtain the token. When complete the Token shows up in your existing Tokens list

Authorization

Headers (4)

Body

Pre-request Script

Tests

Code

Type

OAuth 2.0

Existing Tokens

Get New Access Token

JBAccessToken

Delete

Use Token

Add token to

Header

access_token

eZkMr3u7msiP4VhOF4pvfAggdT

refresh_token

G0mRNjzdvIKHjhORTJWUI6HsEMxMkWyMYicSlad5Q

token_type

Bearer

expires_in

86399

Then you can use the token in your header

Make sure you change “Add token to” to “Header” and then Use Token, which adds it to the header of the Evented Web API request that you are going to build here in Postman

GET create_item Webhook

No Environment

GET create_item Webhooks w/ scope

GET

https://developer.api.autodesk.com/plm360/events/create_item/hooks?scope=urn:adsk:plm:tenant:workspace:PREVIEWEVENTEDWEB.9

Params

Send

Save

Authorization

Headers (4)

Body

Pre-request Script

Tests

Code

Authorization

Bearer 1wwEPig9ZackE9IAPXA708GZLEGg

Bulk Edit

Presets

x-tenant

PREVIEWEVENTEDWEB

Accept

application/json

Content-Type

application/json

key

value



Then you can interact with the Evented Web API

Now we can build our requests to the Evented Web API that will allow us to create a webhook, list all of the webhooks, get information about a specific webhook, or delete a specific webhook. We cover these next.



Learning Objective 3

[Learn about avoiding unique specialized product integrations]

In order to configure the Evented Web API, you will need a Fusion Lifecycle tenant that has Evented Web-enabled. At the time of this writing, the Evented Web API features are only available by request in a Private Beta or Preview tenant. You will use the tenant name to configure your requests to the Evented Web API. You can have the Evented Web send a message (HTTP callback) whenever an item is created in a workspace or whenever a workflow transition is taken. You need to know the workspace ID in order to configure a create_item or workflow transition callback on a specific workspace. You can configure your message to fire for any workflow transition in a workspace, or a specific transition. If you do not specify a workflow transition id through a filter you will receive an event for every transition taken on every item in a workspace. Make sure you really want to do that otherwise you will end up with a lot of noise. Lastly, you need to know the transition ID in order to configure a specific transition.

How to send API requests using Postman

There are four primary requests you can send to the Evented Web API which has 4 distinct URLs. You can POST a Webhook, GET a Webhook, GET a list of Webhooks, and you can DELETE a Webhook. I will cover each of these and their variations (event types, filters) in the following eight examples. The components of your request will include the URL or Endpoint, the Headers, and sometimes the Parameters and the Body. Your endpoint URL is different for each of the following variations, but your Headers are all the same.

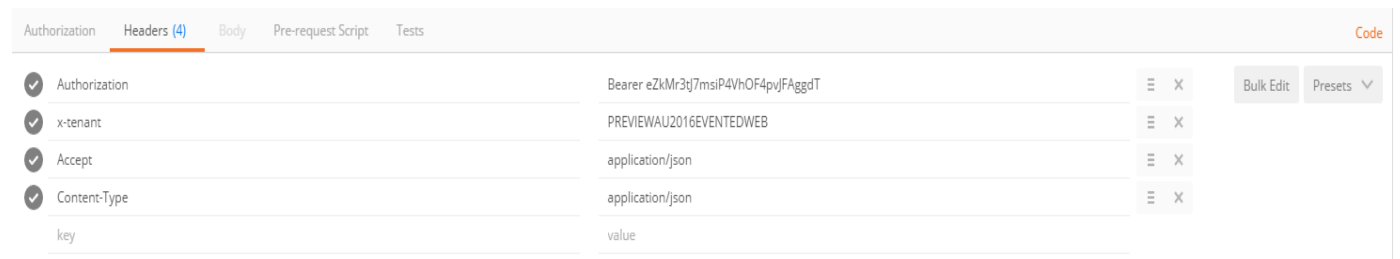
Authorization: Bearer [your access token]

x-tenant: [your tenant name]

Accept: application/json

Content-Type: application/json

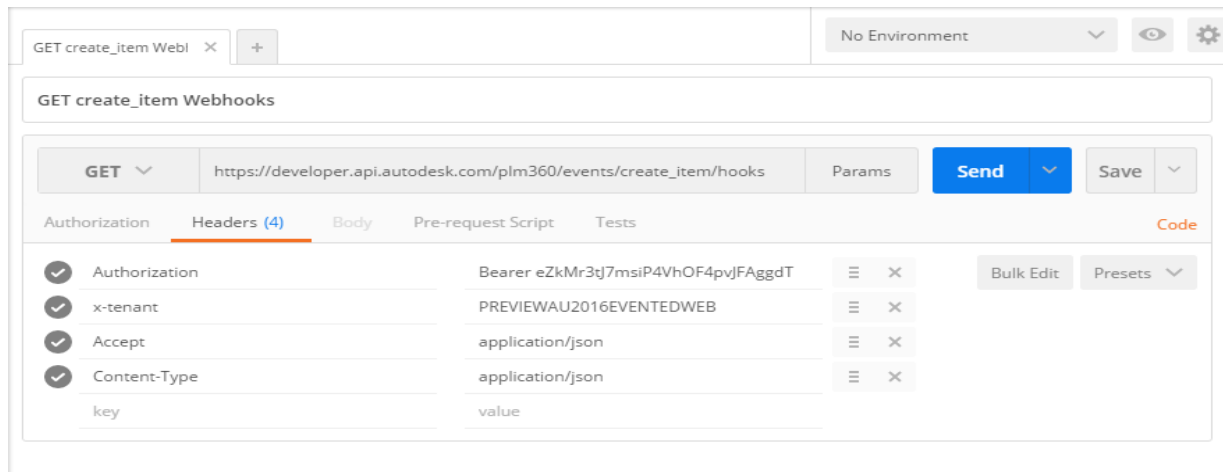
In Postman they look like this



The Authorization access token is applied to your header when you Use Token in a previous example

GET create_item Webhook

The simplest request to send [gets a list of webhooks](https://developer.api.autodesk.com/plm360/events/create_item/hooks) that are already registered on your Fusion Lifecycle Tenant. The Url is https://developer.api.autodesk.com/plm360/events/create_item/hooks and in postman should look like this



Notice the Code button on the right? That is your friend that can write this request in various languages for you, should you ever need to perform this in another system.

For example, here is the cURL

```
curl -X GET -H "Authorization: Bearer eZkMr3tJ7msiP4VhOF4pvJFAggdT" -H "x-tenant: PREVIEWAU2016EVENTEDWEB" -H "Accept: application/json" -H "Content-Type: application/json" -H "Cache-Control: no-cache" -H "Postman-Token: 7e305cd7-e286-72cf-7696-4e6079a0fc51"
https://developer.api.autodesk.com/plm360/events/create\_item/hooks
```

And here is the Javascript XHR

```
var data = null;
var xhr = new XMLHttpRequest();
xhr.withCredentials = true;

xhr.addEventListener("readystatechange", function () {
  if (this.readyState === 4) {
    console.log(this.responseText);
  }
});

xhr.open("GET", "https://developer.api.autodesk.com/plm360/events/create_item/hooks");
xhr.setRequestHeader("authorization", "Bearer eZkMr3tJ7msiP4VhOF4pvJFAggdT");
xhr.setRequestHeader("x-tenant", "PREVIEWAU2016EVENTEDWEB");
xhr.setRequestHeader("accept", "application/json");
xhr.setRequestHeader("content-type", "application/json");
xhr.setRequestHeader("cache-control", "no-cache");
xhr.setRequestHeader("postman-token", "64a9e671-d936-8ccb-de16-2580b5f36277");
xhr.send(data);
```

/ I THINK THAT'S COOL

Send the request in Postman and you receive the response back. If you have a webhook registered you will receive a response like this



GET create_item Webhooks

GET Params

Authorization Headers (4) Body Pre-request Script Tests Code

<input checked="" type="checkbox"/>	Authorization	Bearer eZkMr3tj7msiP4VhOf4pvjFAggdT	<input type="button" value="Bulk Edit"/>	<input type="button" value="Presets"/>
<input checked="" type="checkbox"/>	x-tenant	PREVIEWAU2016EVENTEDWEB		
<input checked="" type="checkbox"/>	Accept	application/json		
<input checked="" type="checkbox"/>	Content-Type	application/json		
	key	value		

Body Cookies (36) Headers (13) Tests Status: 200 OK Time: 1130 ms

Pretty Raw Preview JSON

```
1 {
2   {
3     "active": true,
4     "tenant": "urn:adsk.plm:tenant:PREVIEWAU2016EVENTEDWEB",
5     "callbackUrl": "http://GreenpointTechnologies.jitterbit.net/GreenpointDev/1.0/AU2016.html",
6     "createdBy": "Michael.Pares@greenpnt.com",
7     "createdDate": "2016-11-10T23:04:30.759+0000",
8     "scopes": {
9       "workspace": "urn:adsk.plm:tenant.workspace:PREVIEWAU2016EVENTEDWEB.9"
10    },
11    "_self_": "/systems/plm/events/create_item/hooks/09806370-a79a-11e6-b473-11848a66b403"
12  }
13 }
```

GET create_item Webhook w/ scope Example

Add a scope Param to limit your results to a particular workspace

GET create_item Webhooks w/ scope

GET Params

Authorization Headers (4) Body Pre-request Script Tests Code

<input checked="" type="checkbox"/>	Authorization	Bearer eZkMr3tj7msiP4VhOf4pvjFAggdT	<input type="button" value="Bulk Edit"/>	<input type="button" value="Presets"/>
<input checked="" type="checkbox"/>	x-tenant	PREVIEWAU2016EVENTEDWEB		
<input checked="" type="checkbox"/>	Accept	application/json		
<input checked="" type="checkbox"/>	Content-Type	application/json		
	key	value		

Body Cookies (36) Headers (13) Tests Status: 200 OK Time: 912 ms

Pretty Raw Preview JSON

```
1 {
2   {
3     "active": true,
4     "tenant": "urn:adsk.plm:tenant:PREVIEWAU2016EVENTEDWEB",
5     "callbackUrl": "http://GreenpointTechnologies.jitterbit.net/GreenpointDev/1.0/AU2016.html",
6     "createdBy": "Michael.Pares@greenpnt.com",
7     "createdDate": "2016-11-10T23:04:30.759+0000",
8     "scopes": {
9       "workspace": "urn:adsk.plm:tenant.workspace:PREVIEWAU2016EVENTEDWEB.9"
10    },
11    "_self_": "/systems/plm/events/create_item/hooks/09806370-a79a-11e6-b473-11848a66b403"
12  }
13 }
```



Get workflow_transition Webhook Example

Get workflow_transition webhooks by changing the event type in the URL path

The screenshot shows a REST client interface with a GET request to `https://developer.api.autodesk.com/plm360/events/workflow_transition/hooks`. The request is sent, and the response is displayed in the Body tab. The response is a JSON array containing one webhook object.

```
1 {
2   {
3     "active": true,
4     "tenant": "urn:adsk.plm:tenant:PREVIEWAU2016EVENTEDWEB",
5     "callbackUrl": "https://hooks.slack.com/services/T04F8JC48/...",
6     "createdBy": "Michael.Pares@greenpnt.com",
7     "createdDate": "2016-11-11T17:24:06.345+0000",
8     "scopes": {
9       "workspace": "urn:adsk.plm:tenant.workspace:PREVIEWAU2016EVENTEDWEB.26"
10    },
11    "filter": "$[?@.transitionID == 'urn:adsk.plm:tenant.workflow.transition:PREVIEWAU2016EVENTEDWEB.26.96'",
12    "_self_": "/systems/plm/events/workflow_transition/hooks/a603e390-a833-11e6-8257-832d199fc4d3"
13  }
14 }
```

Get workflow_transition Webhook w/ scope Example

Use a scope parameter to get workflow transition webhooks for a specific workspace

The screenshot shows a REST client interface with a GET request to `https://developer.api.autodesk.com/plm360/events/workflow_transition/hooks?scope=urn:adsk.plm:tenant.workspace:PREVIEWAU2016EVENTEDWEB.26`. The request is sent, and the response is displayed in the Body tab. The response is a JSON array containing one webhook object.

```
1 {
2   {
3     "active": true,
4     "tenant": "urn:adsk.plm:tenant:PREVIEWAU2016EVENTEDWEB",
5     "callbackUrl": "https://hooks.slack.com/services/...",
6     "createdBy": "Michael.Pares@greenpnt.com",
7     "createdDate": "2016-11-11T17:24:06.345+0000",
8     "scopes": {
9       "workspace": "urn:adsk.plm:tenant.workspace:PREVIEWAU2016EVENTEDWEB.26"
10    },
11    "filter": "$[?@.transitionID == 'urn:adsk.plm:tenant.workflow.transition:PREVIEWAU2016EVENTEDWEB.26.96'",
12    "_self_": "/systems/plm/events/workflow_transition/hooks/a603e390-a833-11e6-8257-832d199fc4d3"
13  }
14 }
```



POST create_item Webhook Example

POST a create_item webhook for a specific workspace. You must specify the callbackUrl and the workspace in the body of the request in JSON format as seen here. Toggle the Body to raw for best results.

POST create_item Wel x + No Environment

POST create_item Webhook

POST https://developer.api.autodesk.com/plm360/events/create_item/hooks Params Send Save

Authorization Headers (4) Body Pre-request Script Tests Code

Authorization	Bearer eZkMr3q7msiP4VhOF4pvJfAggdT		
x-tenant	PREVIEWAU2016EVENTEDWEB		
Accept	application/json		
Content-Type	application/json		
key	value		

Authorization Headers (4) Body Pre-request Script Tests Code

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 {
2   "callbackUrl": "http://GreenpointTechnologies.jitterbit.net/GreenpointDev/1.0/AU2016.html",
3   "scopes": {
4     "workspace": "urn:adsk.plm:tenant.workspace:PREVIEWAU2016EVENTEDWEB.9"
5   }
6 }
```

POST workflow_transition Webhook Example

Create a workflow_transition Webhook by changing the event type in the URL and specifying a filter to identify the specific workflow transition id that will trigger the callback.

POST workflow_transi x + No Environment

POST workflow_transition Webhook

POST https://developer.api.autodesk.com/plm360/events/workflow_transition/hooks Params Send Save

Authorization Headers (4) Body Pre-request Script Tests Code

Authorization	Bearer eZkMr3q7msiP4VhOF4pvJfAggdT		
x-tenant	PREVIEWAU2016EVENTEDWEB		
Accept	application/json		
Content-Type	application/json		
key	value		

Authorization Headers (4) Body Pre-request Script Tests Code

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 {
2   "filter": "${?@.transitionID == 'urn:adsk.plm:tenant.workflow.transition:PREVIEWAU2016EVENTEDWEB.26.96'",
3   "callbackUrl": "https://hooks.slack.com/services/",
4   "scopes": {
5     "workspace": "urn:adsk.plm:tenant.workspace:PREVIEWAU2016EVENTEDWEB.26"
6   }
7 }
```



DELETE Webhook Example

When you use GET to return a list of webhooks there is a Guid that gets returned in the response payload under the “__self__” key. Tack that Guid to the end of your URL during a DELETE request in order to Delete a specific webhook. (Event type doesn’t actually matter in this situation)

The screenshot shows the Postman interface for a DELETE request. The top bar indicates the request is in the 'No Environment' state. The main header area shows the method 'DELETE' and the URL 'https://developer.api.autodesk.com/plm360/events/workflow_transition/hooks/89755e10-a53c-11e6-b473-11848a66b403'. Below the URL bar, there are tabs for 'Authorization', 'Headers (4)', 'Body', 'Pre-request Script', and 'Tests'. The 'Headers' tab is selected, showing a table of headers:

key	value
Authorization	Bearer eZkMr3tj7msiP4VhOF4pvjFAggdT
x-tenant	PREVIEWAU2016EVENTEDWEB
Accept	application/json
Content-Type	application/json

On the right side of the interface, there are buttons for 'Send', 'Save', and 'Bulk Edit'.

GET Webhook ID Example

Similarly to DELETE you can GET a specific webhook by tacking its ID onto the end of your request URL

The screenshot shows the Postman interface for a GET request. The top bar indicates the request is in the 'No Environment' state. The main header area shows the method 'GET' and the URL 'https://developer.api.autodesk.com/plm360/events/create_item/hooks/09806370-a79a-11e6-b473-11848a66b403'. Below the URL bar, there are tabs for 'Authorization', 'Headers (4)', 'Body', 'Pre-request Script', and 'Tests'. The 'Headers' tab is selected, showing a table of headers:

key	value
Authorization	Bearer eZkMr3tj7msiP4VhOF4pvjFAggdT
x-tenant	PREVIEWAU2016EVENTEDWEB
Accept	application/json
Content-Type	application/json

On the right side of the interface, there are buttons for 'Send', 'Save', and 'Bulk Edit'.



Learning Objective 4

[Learn how to eliminate complex point-to-point connections]

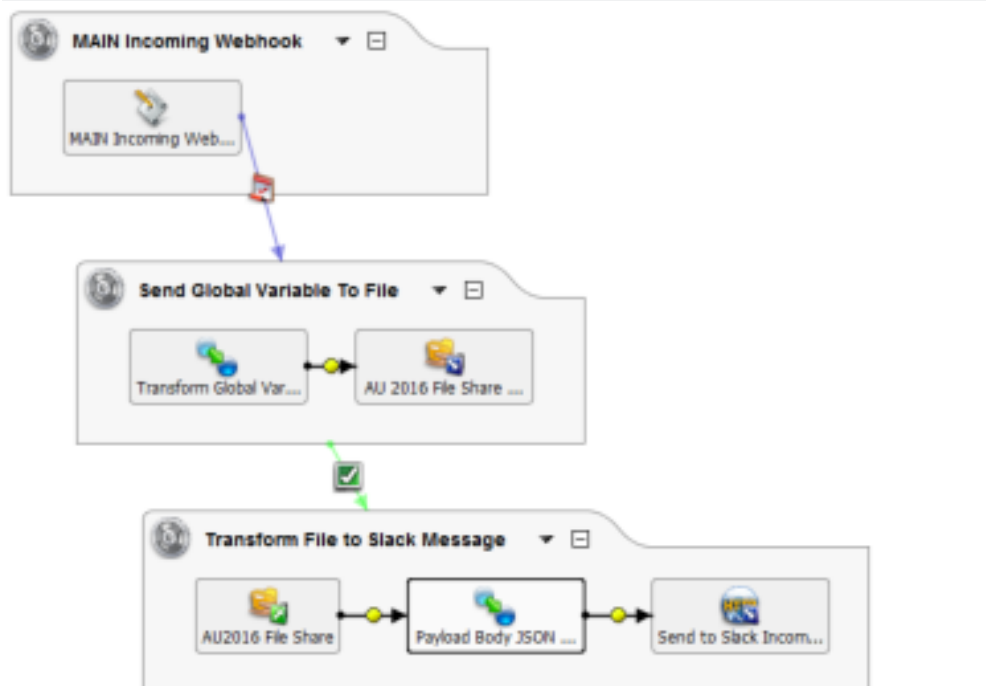
The really fun stuff comes in when you are done with registering and setting up your webhooks. Now you get to put them into practice to do some really cool things! My favorite approach is to leverage Jitterbit Orchestrations, the Harmony API, and the Evented Web to trigger and execute almost any integration my business could need.

Using apis and leveraging Jitterbit - Some examples

So I have registered some webhooks, but what does Fusion Lifecycle actually send out when an event is triggered? The answer is a JSON payload that represents the item that triggered the event. This is great because it's plenty of information to do anything I might want if I just do these three things.

1. Receive the payload at a Jitterbit Harmony API Endpoint
2. Transform the payload so I can use its data
3. Run an orchestration that sends a message to any system I choose

We have already covered creating a Jitterbit Harmony API Endpoint. You simply need to specify the URL for your endpoint as the callbackURL in the body of the JSON payload that you send in a POST request to the Evented Web so that Fusion Lifecycle will send its payload there when an event is triggered. Next let's talk about transforming the Evented Web payload into something you can use for your integration.





Write Evented Web Payload Body to File

Your payload body is stored in `$jitterbit.api.request.body`

We will write that to a file called `request.txt`

The file format is important, make sure you remove the double quote string qualifier so you can store valid JSON in your file. You will consume this file when you transform the contents in the next step.

The screenshot displays the Jitterbit interface for configuring a workflow. The top section, titled "Operations", shows the "MAIN Incoming Webhook" configuration. The "Name" field is set to "MAIN Incoming Webhook". The "Script" tab is active, showing the following code:

```
<TRANS>
$output = $jitterbit.api.request.body;

RunOperation("~/<TAB>Operations/Send Global Variable To File</TAB>");

$jitterbit.api.response = "Acknowledged on: "+Now();
</TRANS>
```

The bottom section, titled "File Formats", shows the "Evented Web Payload Body" configuration. The "Name" field is set to "Evented Web Payload Body". The "Delimiter & Qualifier" section shows "Delimiter" set to "." and "String Qualifier" set to "". The "Validation (Optional)" section shows "Row must contain at least" set to 1 and "columns, if not" set to "Skip Row". The "Define Segment Properties" table is shown below:

Field Name	Type	Default	Format	Validation
body	String			None

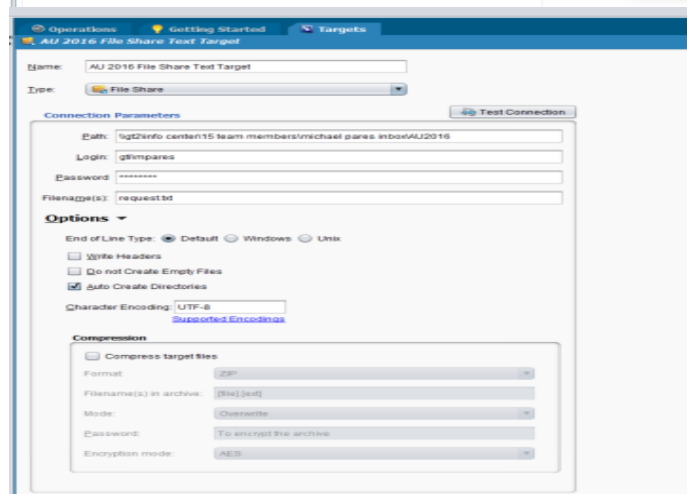
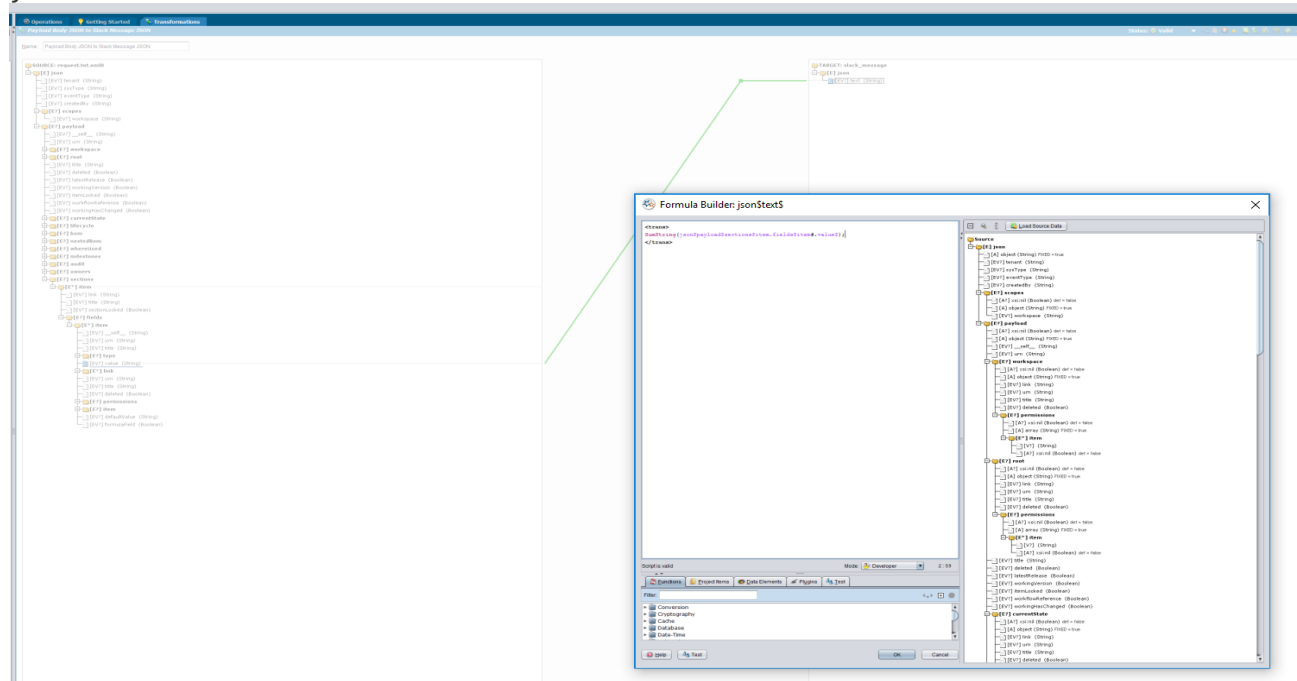
The bottom section, titled "Targets", shows the "AU 2016 File Share Test Target" configuration. The "Name" field is set to "AU 2016 File Share Test Target". The "Type" is set to "File Share". The "Connection Parameters" section shows "Path" set to "lg2info center/15 team members/michael pares info/AU2016", "Login" set to "ghmpares", "Password" set to "*****", and "Filename(s)" set to "request.txt". The "Options" section shows "End of Line Type" set to "Default", "Write Headers" set to "No", "Do not Create Empty Files" set to "No", and "Auto Create Directories" set to "Yes". The "Character Encoding" is set to "UTF-8". The "Compression" section shows "Compress target files" set to "No", "Format" set to "ZIP", "Filename(s) in archive" set to "file.txt", "Mode" set to "Overwrite", "Password" set to "To encrypt the archive", and "Encryption mode" set to "AES".



Transform Evented Web Data into Slack Message

Now you can do whatever you want with your file such as send a message to a slack channel. Let's transform our Evented Web Payload Body JSON into Slack Message JSON format. One thing you may need to do is define your JSON. You can do so when you create your transformation by specifying the JSON file format for the source and target and loading a sample file to create the structure. I downloaded my evented web payload ahead of time from the debug page of the Harmony Web Console, which is easy to do if you turn on debugging on your Harmony API endpoint for the short time it takes to send your API its first payload from Fusion Lifecycle. For Slack, I just typed it because the [slack JSON](#) is so simple

```
{  
  "text": "This is a line of text.\nAnd this is another one."  
}
```





Post a message in Slack

Lastly, let's post our message in Slack. Of course, we could trigger all kinds of other integrations at this point too! I will include those examples in an additional handout.

The screenshot shows a configuration window for a 'Send to Slack Incoming Webhook' operation. The window has a title bar with tabs for 'Operations (3)', 'Getting Started', 'Transformations', and 'Targets'. The main content area is titled 'Send to Slack Incoming Webhook'. It includes a 'Name' field with the value 'Send to Slack Incoming Webhook' and a 'Type' dropdown set to 'HTTP'. Below this is a 'Connection Parameters' section with a 'Test Connection' button. The 'URL' field is set to 'https://hooks.slack.com/services/'. There are empty fields for 'Login' and 'Password'. The 'HTTP Verb' is set to 'POST'. An 'Options' section is expanded, showing 'Content-Type' set to 'application/json' and 'Security Options' set to 'None'. At the bottom, there is a 'Write response to (optional)' dropdown set to 'None' and an 'Advanced Properties...' button.

Closing

The only limit to what we can do with the Evented Web and the Evented Web API is our imagination. As you can see this is a versatile and lightweight solution that addresses many challenges faced by other methods and reduces the cost of entry to integrating all of your business systems. When Integration becomes so inexpensive that we can experience a fundamental change in the way we think, will we shift from asking whether a particular integration is worthy of investment, and instead, ask why on earth we wouldn't integrate two systems when it is only a point and click away?