Field Encryption Setup via Inline or API

Brochure

forcepoint.com

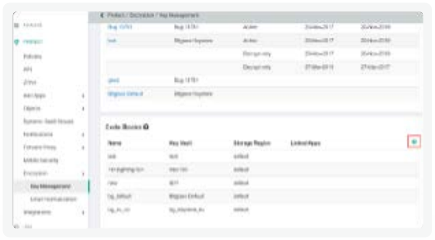
At times you may want to encrypt certain objects or attributes in an application while users input data inline or when passing an API call to an external endpoint. This will require the request to pass through the Forcepoint ONE proxy in order for Forcepoint ONE to encrypt/tokenize the specific objects/attributes in the request. This guide page will walk you through how to add all the components necessary to successfully deploy field encryption in an application.

Setup

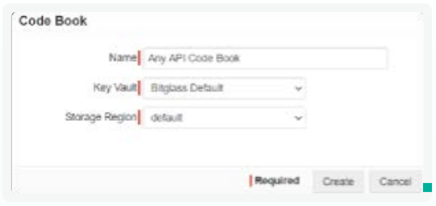
Field level encryption in any app requires a couple of different components to be setup within your Forcepoint ONE tenant. First you will need to add a new codebook for the encryption setup, then add a custom app or a new instance of your custom app where you are moving data to, and then finally add and configure the API endpoint. Before you start you must setup your Key Management key store and key vault ahead of time—either use Forcepoint ONE's default keystore or integrate with your own KMS/HSM. To learn how to do that please view the **Encryption** guide page.

Create the Code Book

1. Start by logging into the Forcepoint ONE admin portal and navigating to Protect > Encryption > Key Management. At the bottom, click the "green plus" icon under the Code Books card to create a new code book.



2. In the Code Book dialog window, enter a recognizable name, select the "key vault" that you created previously that you wish to use, and then select "default" for the storage location.

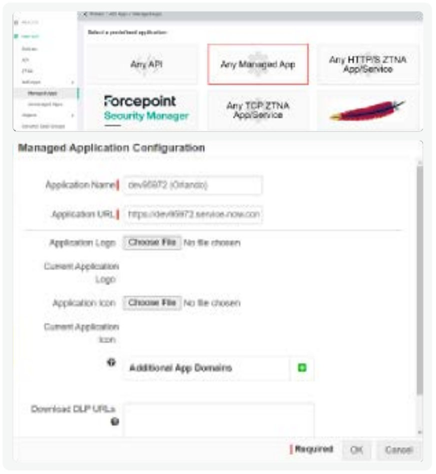


Code Book

Name	Any API Code Book
Key Vault	Bitglass Default
Storage Region	default

Create New Custom App

1. Start by logging into the Forcepoint ONE admin portal and navigating to Protect > Encryption > Key Management. At the bottom, click the “green plus” icon under the Code Books card to create a new code book.


Select a predefined application
Any API
Any Managed App
Any HTTP/S ZTNA App/Service
Forcepoint Security Manager
Any TCP ZTNA App/Service

Managed Application Configuration

Application Name: dev95972 (Orlando)

Application URL: https://dev95972.service-now.com

Application Logo: Choose File No file chosen

Current Application Logo

Application Icon: Choose File No file chosen

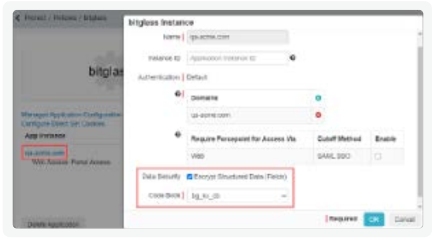
Current Application Icon

Additional App Domains

Download DLP URLs

Required OK Cancel

2. In the Code Book dialog window, enter a recognizable name, select the “key vault” that you created previously that you wish to use, and then select “default” for the storage location.In the Code Book dialog window, enter a recognizable name, select the “key vault” that you created previously that you wish to use, and then select “default” for the storage location.


Protect / Policies / bitglass
bitglass
Managed Application Configuration
Configure Direct Set Cookies
App Instance
qa.acme.com
Web Access Portal Access
Delete Application

bitglass Instance

Name: qa.acme.com

Instance ID: Application Instance ID

Authentication: Default

Domains: qa.acme.com

Require Forcepoint for Access Via	Cutoff Method	Enable
Web	SAML SSO	☐

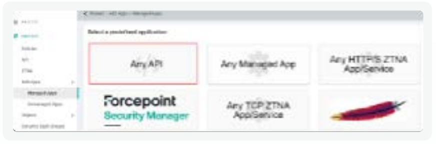
Data Security: ☑ Encrypt Structured Data (Fields)

Code Book: bg_ku_cb

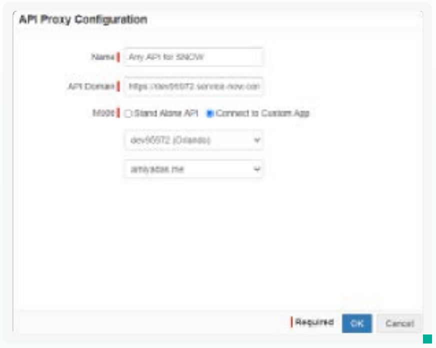
Required OK Cancel

Create Any API Gateway

1. Start by logging into the Forcepoint ONE admin portal and navigating to Protect > Encryption > Key Management. At the bottom, click the “green plus” icon under the Code Books card to create a new code book.


Select a predefined application
Any API
Any Managed App
Any HTTP/S ZTNA App/Service
Forcepoint Security Manager
Any TCP ZTNA App/Service

2. In the API Proxy Configuration dialog, provide a name for the application and enter the URL endpoint you will be using to access the application. Since we are using ServiceNow in our example the URL is “https://dev95972.service-now.com”. Select “Connect to Custom App” and then select the app we created above from the dropdown as well as the app instance (if the custom app you created has more than one). Click “Ok” and then “Save”.



API Proxy Configuration

Name: Any API for SNOW

API Domain: https://dev95972.service-now.com

Mode: ☐ Stand Alone API [✓ Connect to Custom App]

dev95972 (Orlando)

amyadas.me

Required OK Cancel

3. Once done we can now focus on setting up encryption or tokenization for the application. Setup will differ based on if you are planning to encrypt inline, via API, or both.

Encryption/Tokenization Setup

Whether you intend to encrypt or tokenize data inline through the proxy (as users use the app) or via bulk updates over API calls, you will need to add encryption objects under the custom application we created above. Forcepoint ONE provides a number of available encryption/tokenization options depending on your current needs/requirements.

a. **Encrypt:** Encryption uses AES 256 bit encryption. Each data point is uniquely encrypted and stored in the codebook, associated encrypted handle is placed in the field in it's place.

b. **Tokenization:** Tokenization is similar to encryption except it will always tokenize the data and apply the same handle if the data is the same. *For example if you tokenized the name "Acme-gadget" and then later deleted it. When you add that exact name again it will use the same tokenization and associated handle.*

c. **DDM:** Places a marker around the actual text.
For example if the text was "Acme-gadget" then we would store it as "DDMEncrypt (Acme-gadget". If viewed without permission the field will appear blank as opposed to seeing an associated handle. This means the data is not stored in a Forcepoint ONE codebook but remains in the app.

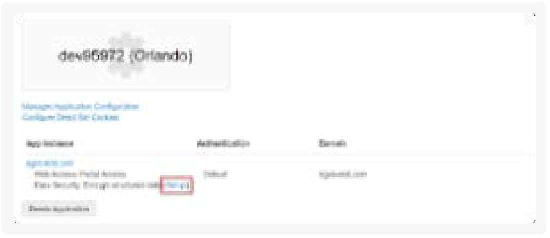
d. **Vaultless Encryption:** Vaultless encryption is unique in that Forcepoint ONE will encrypt the text and place the newly encrypted text back into the application. Since we are placing the actual encrypted text and not a handle, you must set the "Max Length" field to at least 10x the actual characters you would support in that field. *For example if you were encrypting an address field that supported up to 50 characters then you would need to input 500+ for the "Max Length.*

e. **Vaultless Local Tokenization:** Vaultless Local Tokenization works similarly to vaultless encryption in that we will tokenize the plaintext and place the ciphertext back into the application. If multiple form paths are mapped to the same Object. Field and we see the same plaintext within a single request (even across different form paths) then we will tokenize the first instance of the plaintext with a tokenized ciphertext and then use that same ciphertext for each other instance of that plaintext that we see within that request. For new requests or requests in different Object.Fields we will create new ciphertexts.

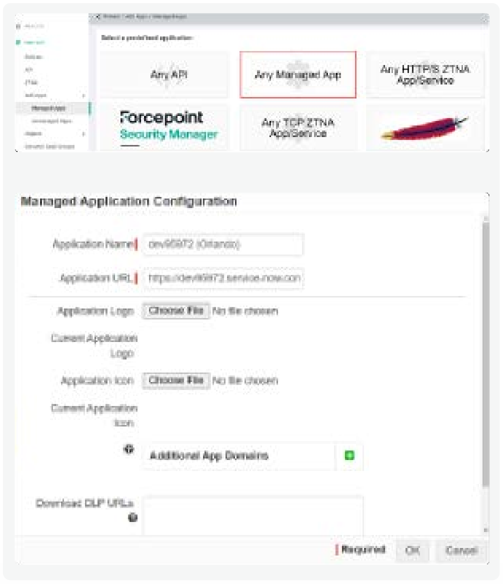
→ *For example if you have path1 and path2 configured for Object.Field1 and you sent a single request through with plaintext "abc" appearing in both path 1 and path 2 then Forcepoint ONE will tokenize the plaintext "abc" with a ciphertext and use that same ciphertext for both instances of "abc" in path1 and path2. If you send a new request with "abc" through path 1 and path 2 then Forcepoint ONE will tokenize "abc" with a new ciphertext and use that for both instances. If in a single request*

→ *If you sent "abc" to path1 and path2 in Object. field1 and also to path3 and path4 in Object. Field2 in the same request then "abc" will be tokenized once and given the same ciphertext for path1 and path2 in Object.Field1 and then a different ciphertext will be used for path3 and path4 in Object.Field2.*

To begin we need to navigate back to the custom application we created above and click on the "Data Security: Encrypt Structured Data" option.



1. Under Encryption Summary, click the "green plus" icon to create a new object. You will need to create at least 1 object which will be the primary key. Provide an "Object Name", a "Field", "Type", "Max Length", "Action", and "Security Level". You can add additional objects once you have set the primary key to specify actions you will want to take on the data. There are a few caveats when adding the objects:



Managed Application Configuration

Application Name	dev95972 (Orlando)
Application URL	https://dev95972.service-now.com
Application Logo	Choose File No file chosen
Current Application Logo	
Application Icon	Choose File No file chosen
Current Application Icon	
Download DLP URLs	

   a. Object needs at least one field that's marked as a primary key.

   b. For the primary key you cannot select "encrypt" as an option, only "tokenize" or "none (plain text)". This is because the key will be used by both the endpoint and the end service and must be recognized as the same. Encryption would cause the ciphertext to be different each time.

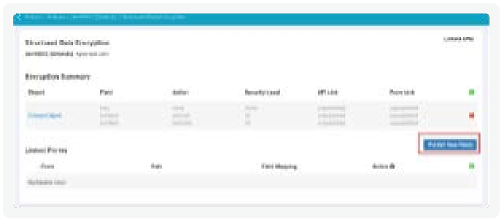
   c. A field needs to be at least 40 characters to be able to select encrypt or tokenize for the action.

   d. While you must set a "Security Level" if you select tokenize, the number you enter does not matter for the primary key. "Security Level" will be used when encrypting the data in order to control user access or ability to view the data.

2. Now you can add new objects to specify the action you will be taking – for either inline or via API in the later configuration – and what security level you want to set. The security level number is arbitrary and you can choose to set it to whatever. When configuring policy you will denote what security levels users will get based on the policy line they hit.

   → *For example if you set the action to encrypt at a level 10 for an object key and another encrypt at a level 20. When you apply these different encryption levels to the data, users can be assigned specific security levels based on policy line they hit. An admin on a managed device can get a higher level (say 20 or more) while a user on an unmanaged device will get a lower security level (10 or less). This will control what data they will be able to view/ access*

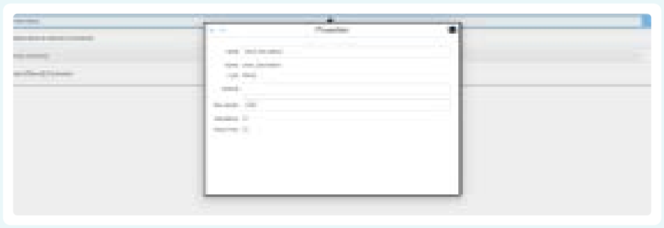
3. After configuration click “Save” and then “Publish Fields” back on the Structured Data Encryption page.In the



Notes

→ You can add as many encryption objects as needed to separate the types of encryption/tokenization used later for inline or API protection.

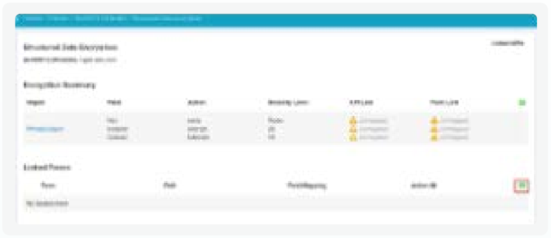
→ For ServiceNow you are able to configure/set the character length for any of the fields you are using. You will need to make sure that the max field length you set in Forcepoint ONE is equal to or less than the field length you set in ServiceNow since Forcepoint ONE will not be able to place the encrypted text back into the field if the character length exceeds what the field can hold.



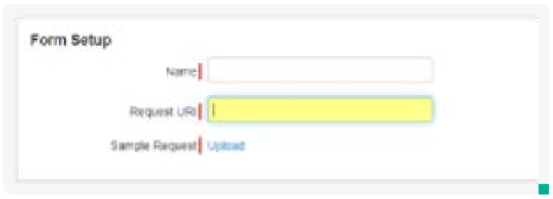
Inline Encryption

If you do not intend to perform inline encryption and only encrypt via API calls you can skip this section.

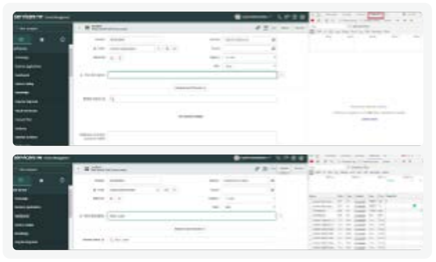
4. Now we need to learn the fields and map the linked form. Under “Linked Forms” click green plus icon.



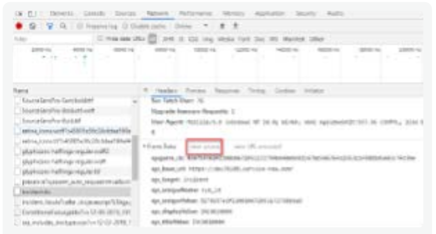
5. Provide a name for this form setup. Again this name is arbitrary but should be something to help you identify what it is for. Leave the "Request URI" blank for now as we will discover it shortly. Click "Edit Sample Request" and we now need to find the "Request Body" in the application to discover the fields that we will be making changes to.

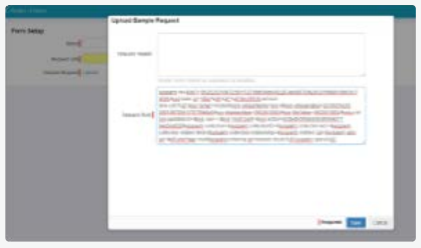


a. To figure out the "Request Body" we will need to do a network capture inside the app itself attempting to upload or post to a field on the page you wish to make changes to. For this example we will be looking at "Incidents" in Servicenow. While sitting on an "Incident" page open up the browsers "Inspect" function. Go to the "Network" tab at the top and clear out any traffic to start clean. Fill out whichever field you wish to control and click submit.

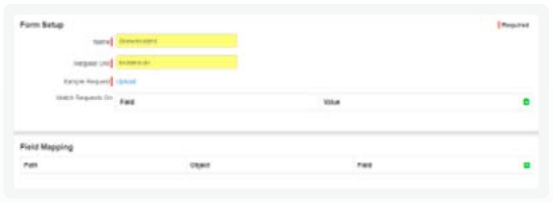


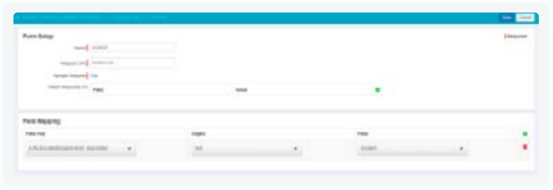
b. In the "Network" tab we need to search for the post. In this case it is "incident.do". This will also be the "Request URI", enter that inside of the Forcepoint ONE portal. Select "incident.do" scroll down to "Form Data" and select "view source". Then copy the contents (make sure to scroll to the bottom and "show more" to get the entire content) and paste it into the "Sample Request" in Forcepoint ONE.



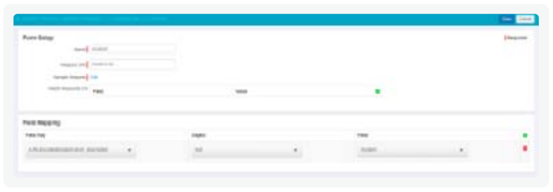


6. Now with the "Name" and "Request URI" fields filled out and the sample request uploaded you should now see an additional section at the bottom for field mapping. This will allow you to select the available fields that we saw within that form based on the "Sample Request" uploaded.





7. Now you can add fields to encrypt. Select the field key that Forcepoint ONE discovered (here we are choosing short description). Select the Object and Field that you had created back in step 2 of this encryption setup. Remember you can have multiple objects/fields that you created each with different levels of encryption/tokenization. Here we only created one.
   Now this means I will have encrypted the "Short Description" field with a level of 20 based on my configuration above.



8. Once done you can now check to make sure that data that is entered into that field through the proxy is tokenized or encrypted based on your security level. This means users going direct or without the proper security level will not see the data in a decrypted state.

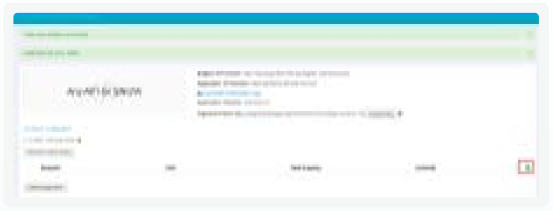
   Notes

   When configuring contextual policies to decrypt data and using custom locations, if you have the same IP address in multiple custom location objects it is possible that one of those locations is configured for a policy where users are not allowed to view data. In this case the user will not see data in a decrypted state coming from that IP regardless of policy. Forcepoint ONE recommends avoiding duplicate IP addresses or ranges across multiple custom location objects. To learn more about custom location objects please view the Objects guide page.

Any API Field Encryption

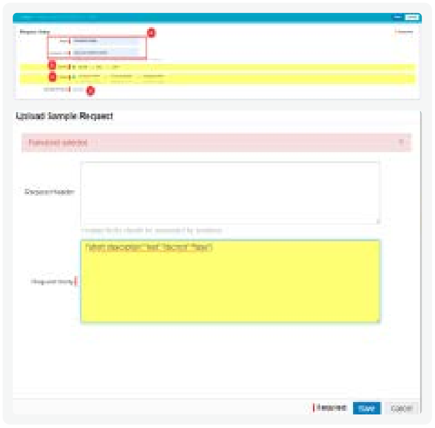
The API encryption and field mapping will instead be performed in the Any API app we added above.

1. Back on the API application settings page click the "green plus" icon to add a new Endpoint.

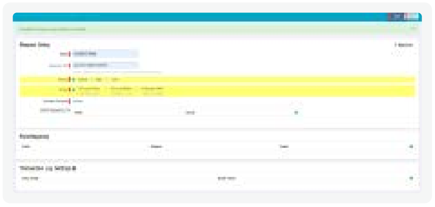


   a. First provide a name and enter the request URI.
   b. Select the format which you will be sending data for this endpoint (JSON, XML, CSV).
   c. Select the action that you are taking (inline, batch, decrypt).

d. Click the "Sample Request" option and enter an example of a data request that will be uploaded using this API and click "Save" at the bottom.



e. Click "Save" at the top right of the "Request Setup" page.

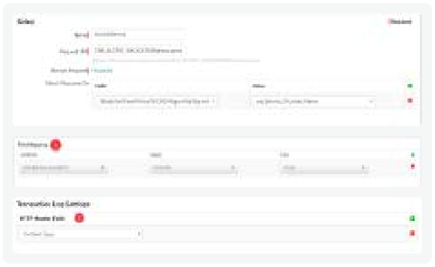


2. After uploading the "Sample Request" you are presented some new fields to configure:

a. Match Requests On: This table allows you to select which field and value needs to be present before performing any actions. Click the "Green Plus" icon to add a new line and select the "Field" from the dropdown and select the appropriate "Value" that needs to be present before you take actions. If an API request contains a field name such as "Content Type" and values of "Account Name", "Client Name", and "Social Security Number" then the actions configured on the "Actions Taken on Matched Requests" card will be applied. If no fields are configured in the "Match Requests On" table, then the actions will be applied during all API requests based on the selected fields in the "Actions Taken On Matched Requests" card. In my example I have input a field of "decrypt" with a value of "false". Meaning if I include that field and value in my post then the data I am pushing will follow the action in the field mapping (in this case encrypt). If I don't include that then no actions will be taken.

b. Field Mapping: This card will let you select which particular fields will be encrypted or tokenized. Click the "Green Plus" to add a new line and select the "JSONPath" you wish to take an action on. Then select the "encryption object" you created above in step 2 of the main encryption setup as well as the "field" to specify the exact action being taken. Again actions will only be taken if a particular value is matched based on the "Match Requests On" table. If nothing was filled out for "Match Requests On" then your configured fields will always be encrypted or tokenized in your API requests.

c. **Transaction Log Settings:** Here you can specify a "HTTP Header Field" that will appear in your logs and will allow you to track and/or debug requests. You can add as many header fields as you would like to add them as items that can be tracked and filtered in your Dashboard logs.

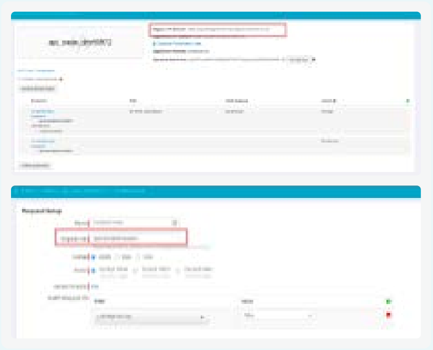


3. Once you have configured the field mapping you are now set and can move on to pushing data via the API and having it be encrypted or tokenized as desired.

Notes

When making API calls you must ensure the following two items:

1. The API endpoint you are posting to equals the Forcepoint ONE API Domain URL you find in the API app settings appended by the API endpoint URI you are trying to post to. For example with our app we created the URL would be "https://bg-u9mbigyfzk-874.api.bitglass.onpremise2.net" and the endpoint URI is "/api/now/table/incident/" so combined we would get "https://bg-u9mbigyfzk- 874.api.bitglass.onpremise2.net/api/now/table/incident"



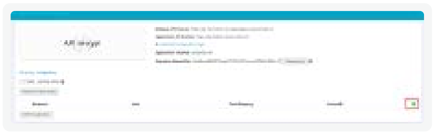
2. You must also ensure that your API calls include a custom Forcepoint ONE header "X-BG-SIGV2" with the value of the header being the signature key generated on your API app. Please see the example Postman calls below if you would like to see an example of data being pushed through this API flow to be encrypted in ServiceNow.

Decrypt Flow

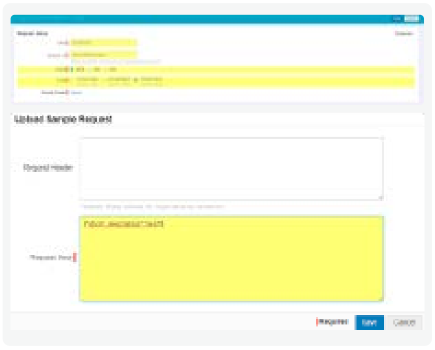
At times data that is encrypted by Forcepoint ONE in an app is moved to another app. Data will need to be decrypted before it is sent or else the new application will not be able to interpret the data. To do so you will need to create a new API app to setup a decrypt endpoint so encrypted data can move through the Forcepoint ONE API proxy and be decrypted.

1. To begin we will need to add another API app. Follow the instructions under the Create Any API Gateway section above.

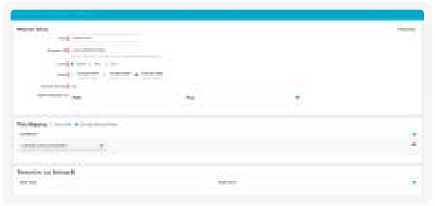
2. In the API app settings add a new endpoint by clicking the green plus icon



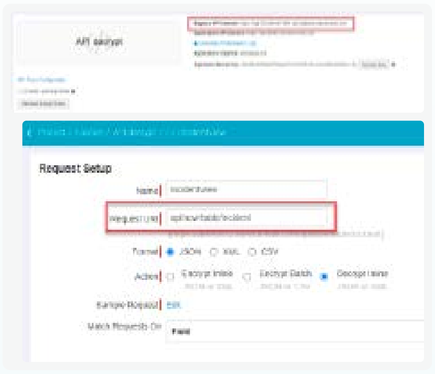
3. Provide a "Name", enter the "Request URI" data will be moving to, Select the "Format" (should be JSON or XML), select "Decrypt Inline and then click "Upload" to enter a sample request.



4. ou can add "Match on Requests" if you'd like it to only decrypt for specific posts, otherwise leave blank and just configure the field mapping with the specific JSON Path requests you want decrypted.



5. Now you have your decryption API endpoint. When moving data from an app with encrypted data to another app make sure you are using this new apps endpoint domain URL similar to above. For this example ours would be "https://bg-7j7ct34mj7- 964.api.bitglass.onpremise2.net/api/now/table/incident" combining the Forcepoint ONE domain URL given to us on the settings page happened with the Request URI.



api/now/table/incident
Format: [✓ JSON] XML CSV
Action: Encrypt Inline, Encrypt Batch, [✓ Decrypt Inline]