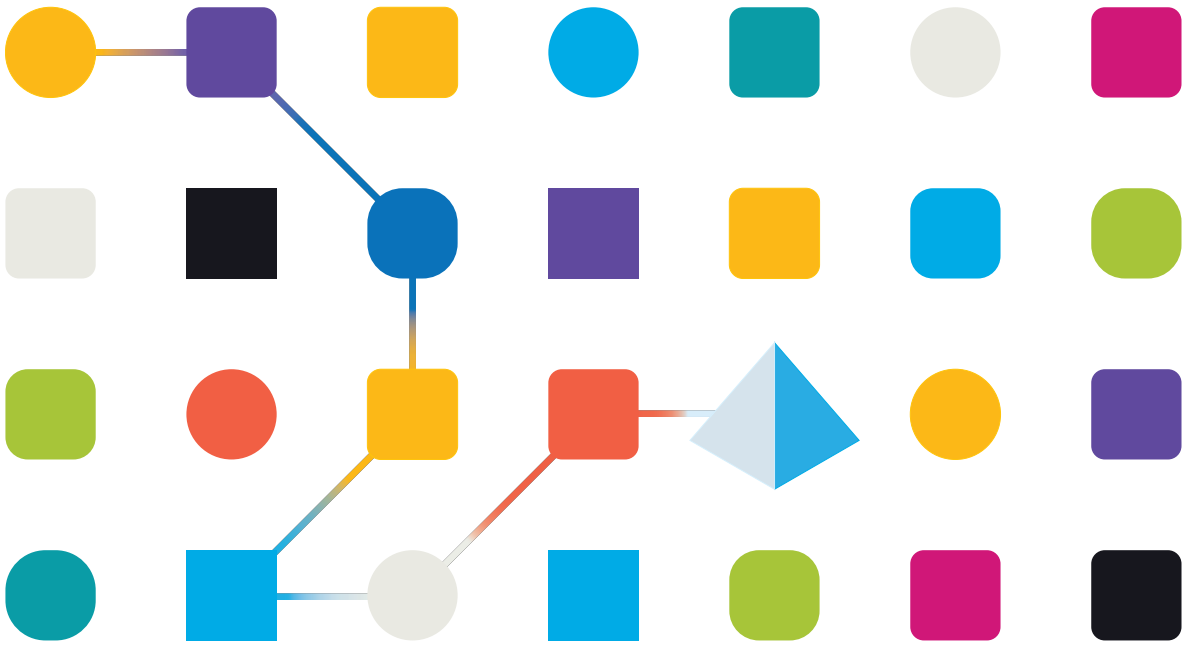
Blue Prism 7.0

Web APIs

Document Revision: 3.0



Trademarks and Copyright

Blue Prism Limited, 2 Cinnamon Park, Crab Lane, Warrington, WA2 0XP, United Kingdom.
Registered in England: Reg. No. 4260035. Tel: +44 370 879 3000. Web: www.blueprism.com

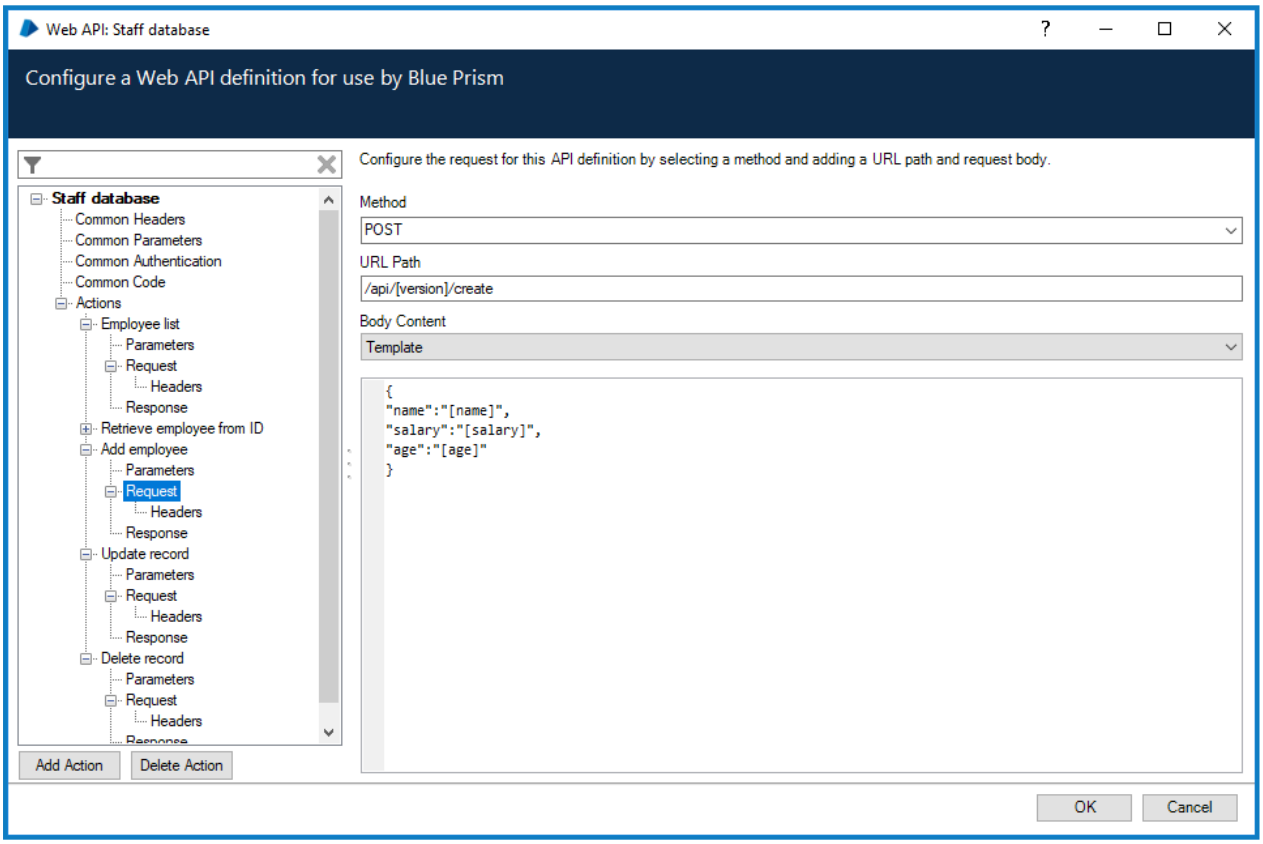
Contents

Web APIs

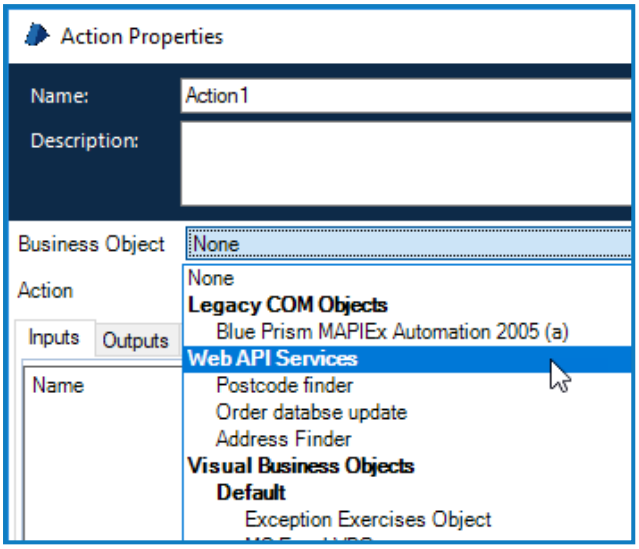
The Web API functionality provides an interface for configuring native interactions with systems and services that provide published HTTP APIs – the most common of these is RESTful web services.

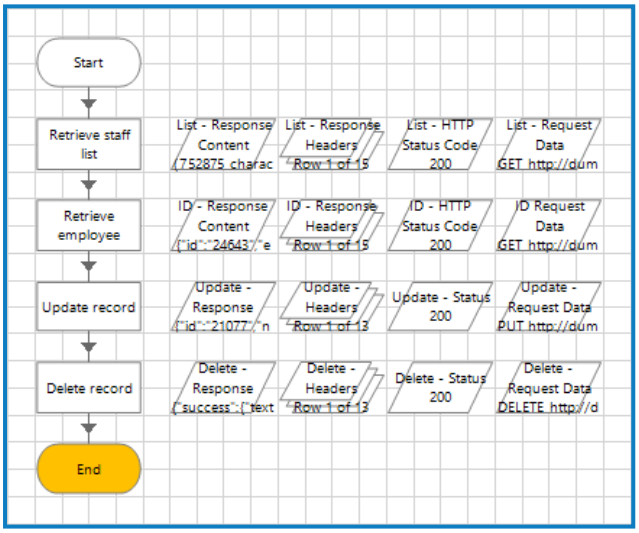
The Web API Services feature allows Blue Prism processes to interact with these services to either provide data to, or to consume the data or services provided by these external systems within an automated business process. The features provided natively by the Web API functionality allow the most common services to be automated by Blue Prism and these capabilities can be extended using code stages to cater for bespoke or complex data structures and authentication mechanisms.

The Web API configuration tool has been designed to be as flexible as possible to ensure that complex API calls can be configured. Web API definitions are created in the Blue Prism System tab and each definition can contain multiple actions, each one typically representing an endpoint within an API. The actions define how an HTTP request is made and determines how the HTTP Response is translated into output parameters.



Once an API definition has been configured, the associated Actions can be used in Objects to enable Process Developers to use interactions with third-party services as part of an automation.





Create a Web API definition

Web API definitions contain all the data required to instruct Blue Prism how to make HTTP requests to a Web API. The data in the API definition determines how each request is constructed, specifying the required URL, HTTP method, headers, and authentication required to make a request. API definitions also allow parameters to be defined that can be used to pass data into a request.

Each API definition is made up of the following elements:

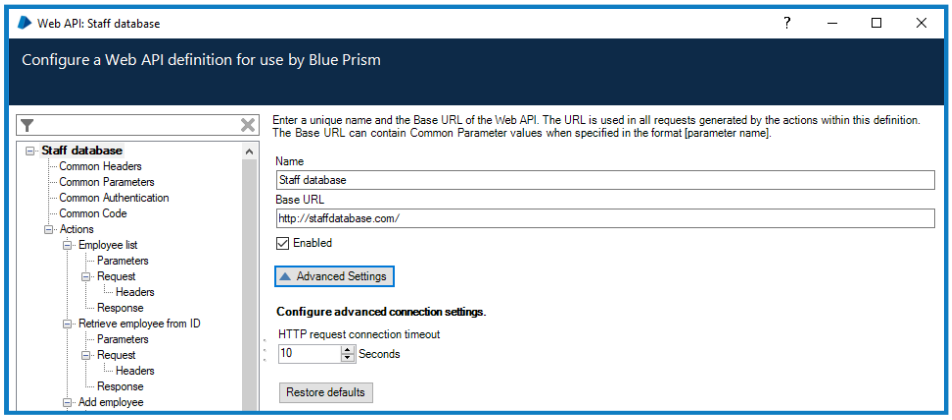
- Basic settings – The name, base URL, and timeout settings used for all API requests configured in the API definition.
- Actions – The capabilities provided by the Web API that will be made available to Blue Prism. The actions define the request body and response handling for each API request in the definition.
- Headers – The headers that will be used in the requests for this API definition. Common headers are used in all requests, and action-specific headers are used, in addition to any common headers, for the actions in which they are defined.
- Parameters – Dynamically expanding references that are used to pass data into base URLs, URL paths, headers, and request bodies. Parameters can also be used in the *Subject* field for *OAuth 2.0 (JWT Bearer Token)* configurations. Common parameters can be used in any action, action-specific parameters can only be used in the action in which they are defined.
- Common authentication – How each request to an API is authenticated.
- Common code – Code used to create complex API requests and responses.

> The Web API Action stage expects a successful response from the API request. Any unsuccessful requests will throw an exception which should be handled accordingly. For more information, see Error handling.

Web API definitions are configured in the System settings of the Blue Prism interactive client.

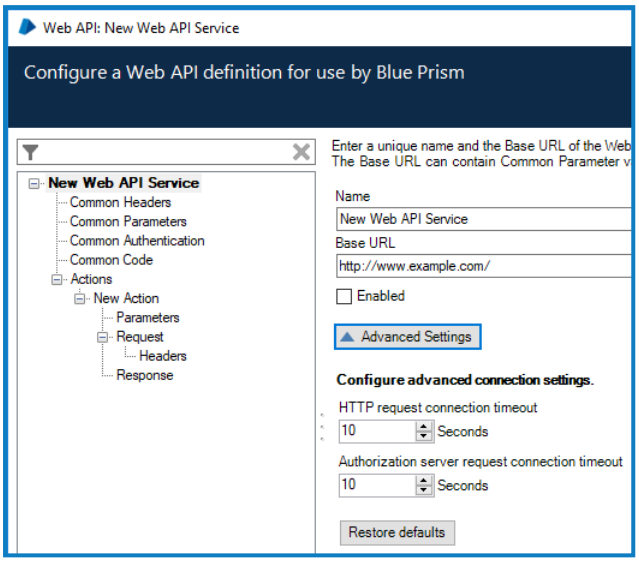
1. Select the **System** tab.
2. Select **Objects** > **Web API Services** from the navigation tree.
3. Click **Add Service** from the options menu.

   The New Web API Service dialog displays.



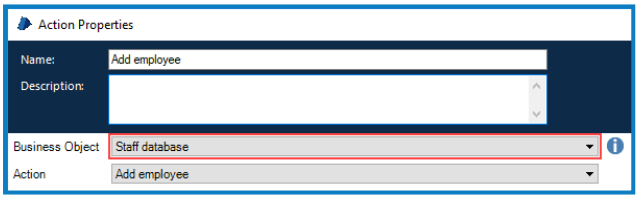
Basic settings

The common components and settings for all requests in an API definition.



Name

A name, unique in the environment, that is used to identify the API definition in Blue Prism. The name is used when selecting the API definition in Action stages.



Base URL

The part of a URL that is used for every HTTP request within the API.

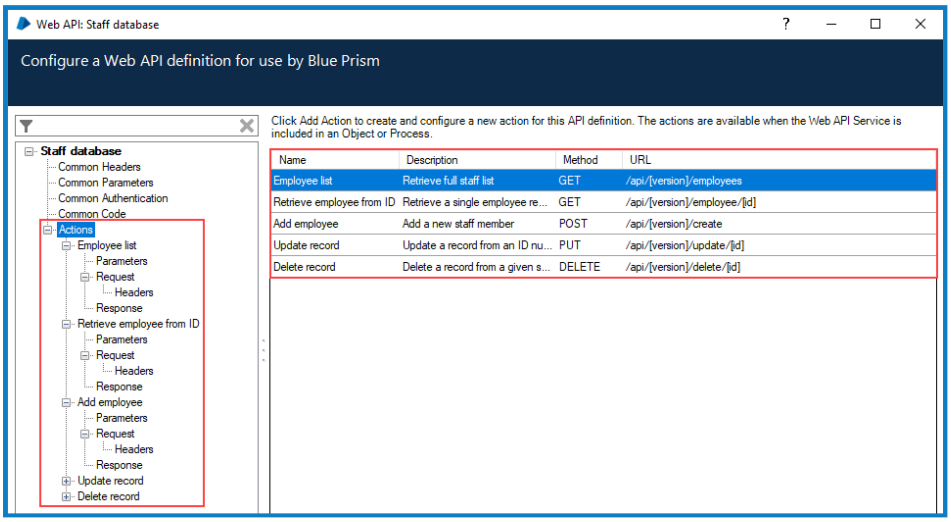
Advanced settings

Apply connection timeout settings for configured web services:

- **HTTP request connection timeout** – The maximum length of time that Blue Prism waits for a response to an HTTP request before an exception is thrown.
- **Authorization server request connection timeout** – The maximum length of time Blue Prism waits for a response to an authorization server request before an exception is thrown. This setting is only available for OAuth 2.0 (Client Credentials) and OAuth 2.0 (JWT Bearer Token) authentication types when selected in the Common Authentication settings for an API definition.

Actions

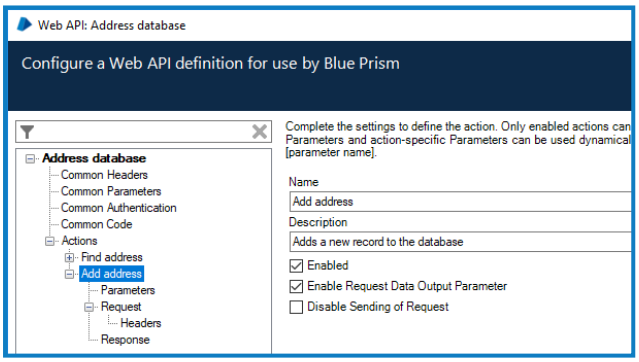
A Web API definition can include multiple Actions, each one representing a different request to a specific endpoint using the required HTTP method. All configured actions for an API display in the navigation tree and are listed in the main screen when the **Actions** node is selected.



Click **Add Action** to create a new action for an API definition and configure the associated parameters.

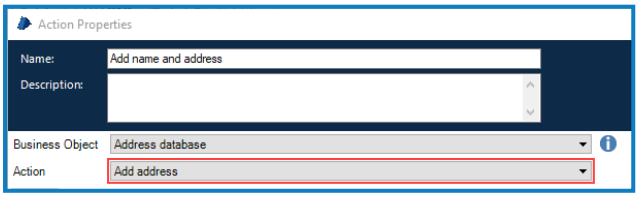
Basic API definition settings

Add a name and description and apply action specific settings.



Name

A unique name for the action. The name is used when selecting business objects for actions in Process and Object Studio.



Description

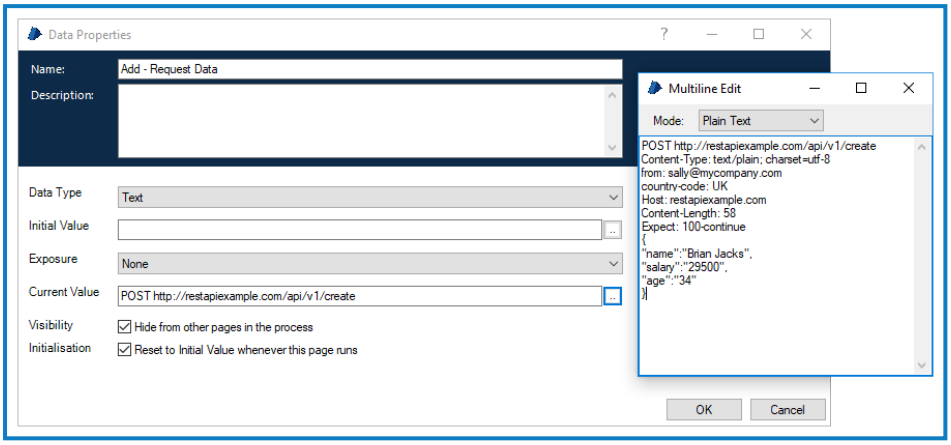
An optional description to provide more information about the action.

Enabled

Actions must be enabled to be available for selection in objects and processes.

Enable Request Data Output Parameter

When selected, creates a *Request Data* output for an action in Process and Object Studio. When an action runs, the associated data item is populated with the API request data.

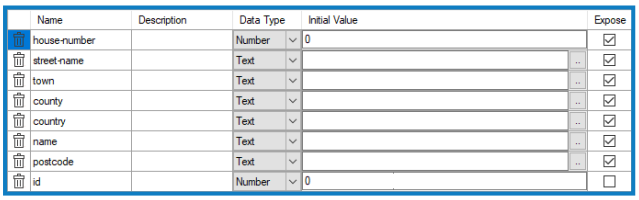


Disable Sending of Request

When this option is selected, the request data is generated but the request is not sent to the server. Use in conjunction with *Enable Request Data Output Parameter* to generate a request without contacting the server, providing a mechanism for debugging and evaluating requests.

Parameters (action-specific)

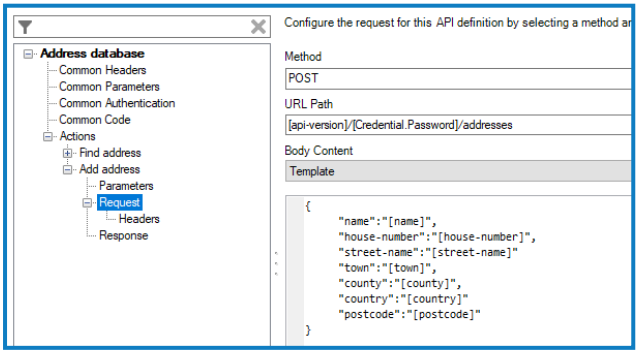
These parameters are specific to the action and are used in combination with the common parameters as the inputs for an action in objects and processes. When enclosed in square brackets, parameter names are used dynamically to add their values to URLs, header values, and request bodies.



For more information, see Parameters.

Request

Define the HTTP Request that will made to this API endpoint by selecting the required HTTP method and adding the URL path, body content, and headers.



Method

The HTTP method used for the request. The drop-down contains the standard HTTP methods but any request verbs can be entered into the free-text field. Validation is performed on the field to prevent the input of non-allowed characters.

URL Path

The URL path particular to the action that defines the specific request when used with the base URL. The URL path can be parameterized using common and request-specific parameters, enclosed in square brackets.

For example, the base URL is:

```
http://staff.database.com
```

The URL Path is:

```
/api/[version]/employee/[id]
```

When the parameters are expanded and added to the base URL, a request is made to this address:

```
http://staff.database.com/api/v1/employee/26855
```

Body Content

Configure the content to be sent with the HTTP request using one of the following content types:

- **None** – The request does not include body content. This is usual for actions that retrieve information, such as those using the GET method.
- **Template** – Send text-based body content such JSON or XML. Parameters can be used to add data to the request body.
- **Single File** – Send files as part of the API request. The API definition specifies an input parameter that is used to reference a file from a data stage in a process.
- **Multiple Files** – Send multiple files as part of the API request. The API definition specifies an input parameter that is used to reference files from a collection stage in a process.
- **Custom Code** – For complex scenarios the request body can be generated using code. Where required, this can incorporate code from the Common Code area of the API definition.

Headers (action-specific)

Add a name and value for each header specific to this action. The headers used in a request are a combination of the common headers in the API definition and the headers set at action level. If an action-specific header has the same name as a common header, the value of the action specific is used in any requests.

For further information, see Headers.

Response

Transform the HTTP response using JSONPath or custom code to extract the required data. The data can be stored in data items, making it easier to use in other stages of an object or process. Custom code can be used to configure more complex responses, for example, performing calculations on the response data.

The response output is defined using the following fields:

- **Parameter** – The name of the output parameter created to store the data specified.
- **Data Type** – The data type of the output parameter.
- **Method** – The method can be either **JSON Path** or **Custom Code**. If using **Custom Code**, the outputs are ref parameters that need to be assigned.
- **JSON Path** – If using **JSON Path**, the expression needs to extract specific elements of the JSON into output parameters. These output parameters have a specified Blue Prism data type, and the object found using the JSONPath will be converted to that data type.

> For further information about configuring responses, see Extract response data with JSONPath.

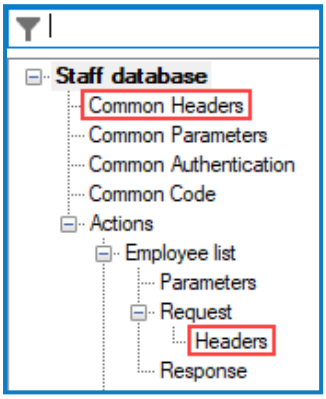
Error handling

If the HTTP request times out, or the response status code indicates a problem with the request (4xx) or a server error (5xx), an exception will be thrown and the Web API Action stage will not produce output parameters. When using a Web API Action stage, the process designer should handle these exceptions using the existing error handing mechanisms (Block, Recover, and Resume). The error message generated from a Web API exception is written in a standard format, containing the HTTP response status, headers, and body. The process designer can parse this message and add logic to handle specific HTTP status codes as required.

Headers

Configure the HTTP headers to be sent as part of a Web API request. Headers can be defined for the whole API and for individual request methods:

- **Common** – Common headers are sent in every request generated by the actions in an API definition.
- **Request-specific** – Headers that are only sent in requests for the associated actions.

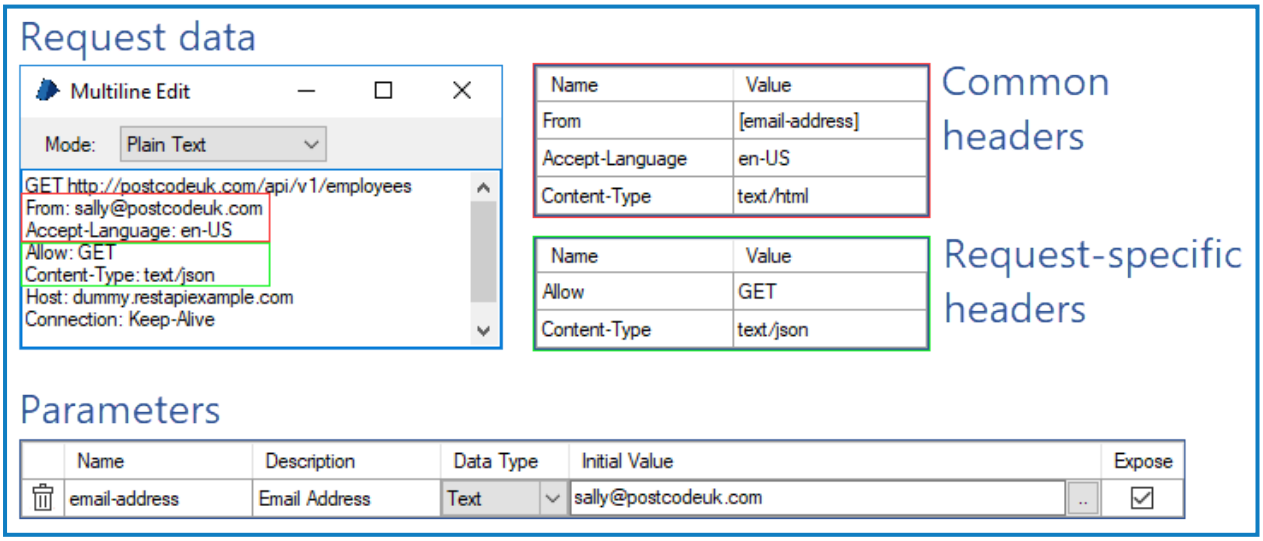


Headers are defined by entering a name and the required value. Values can include parameters, that dynamically add their values when the API is called.

The same name can be used for common and request-specific headers. Where this is the case, and both have a value configured in the API definition, the value of the request-specific header is used when the API is called.

This example shows an API request, created from an API definition:

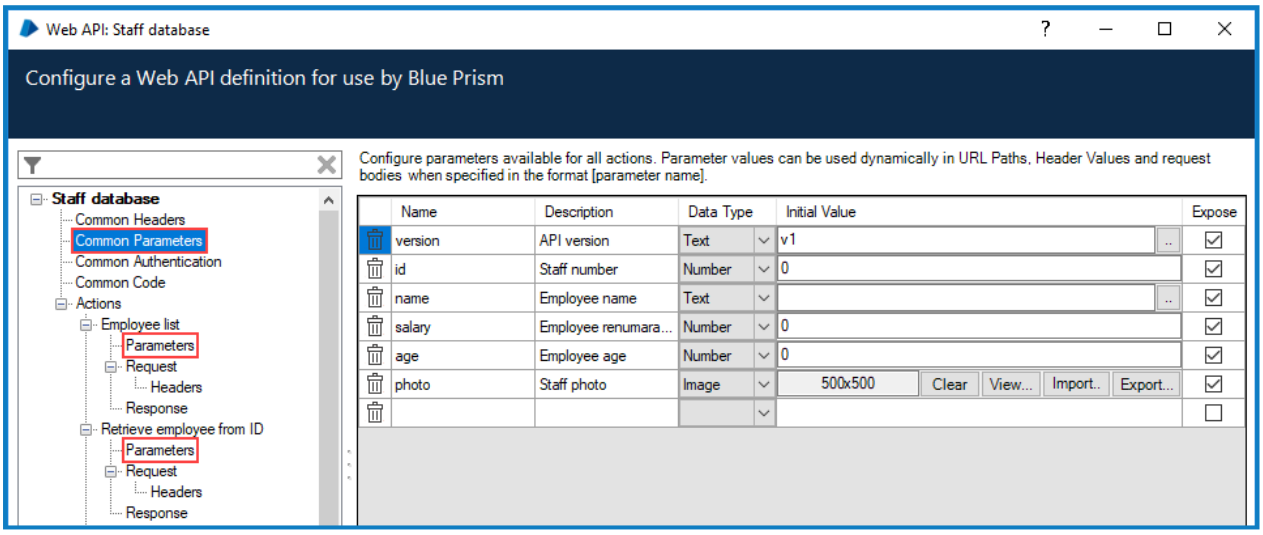
- The request includes both common and request-specific headers.
- The *Content-Type* header has a value set in the common header that is overwritten by the value in the request-specific header when the API is called.
- The *From* header is populated with the value set in the *email-address* parameter.



Parameters

Parameters provide a method of passing data into an API request by dynamically adding a value when an API is called from within a process. They can be used in base URLs, action URL paths, headers, request bodies, and when exposed to processes, they are available as inputs for actions in objects and processes when the appropriate API service is selected as a business object.

Two types of parameters can be configured in an API definition, common parameters and action specific parameters. Common parameters are used in all actions and parameters configured at action level are available only to that action.



To use a parameter value in URL paths, headers, and request bodies, enclose parameters in square brackets. For example:

```
http://staff.database.com/api/[version]/employee/[id]
```

When a process that uses the associated API definition is run and the parameters are expanded, the URL could become:

http://staff.database.com/api/v2/employee/290054

To use square brackets without referencing a parameter, the text must start with double square brackets. For example, if the request body included:

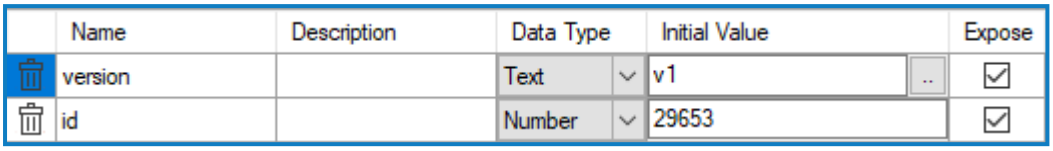
```
{
"favourite-colours": [["green", "red", "blue"]
}
```

When the API is called, the HTTP request includes the square brackets:

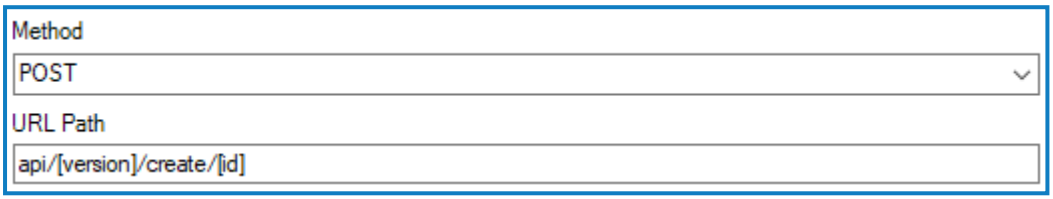
```
{
"favourite-colours": ["green", "red", "blue"]
}
```

Parameters are applied at the point at which an API is called and values are taken from the API definition if an initial value has been set. However, if the parameter has been exposed and a value has been entered as an input in a process action, this value will be used.

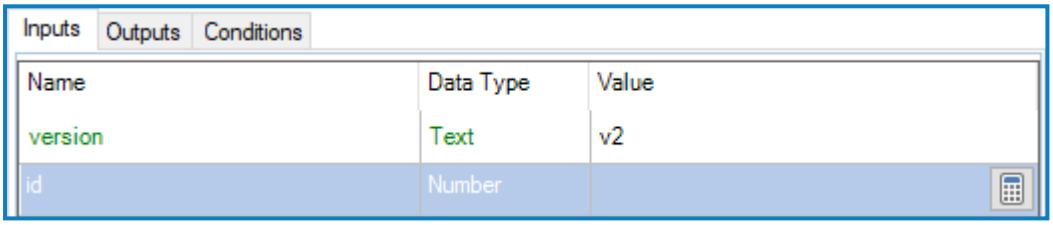
In the following example, two common parameters are configured in the API definition, both with initial values set and both exposed to processes.



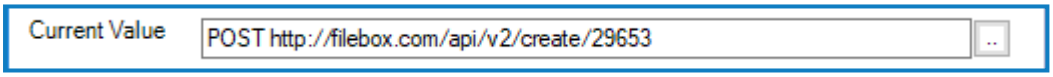
The parameters are referenced in the request URL.



An input value for the *version* parameter is set in an action but not for the *id* parameter.



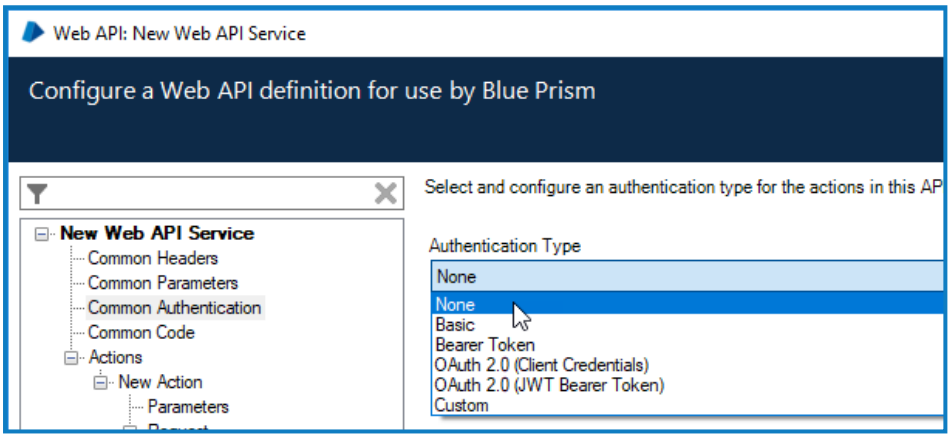
When the action runs and the API is called, the *version* parameter value is taken from the API definition and the *id* from the action input.



Common authentication

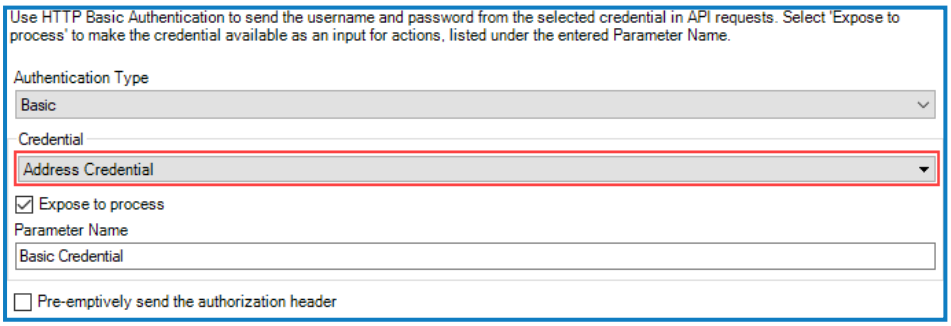
Where API requests require authentication to send and receive information, credentials specific to the authentication type can be configured. Blue Prism supports the use of the following authentication types for API requests:

- Basic
- Bearer Token
- OAuth 2.0 (Client Credentials)
- OAuth 2.0 (JWT Bearer Token)
- Custom

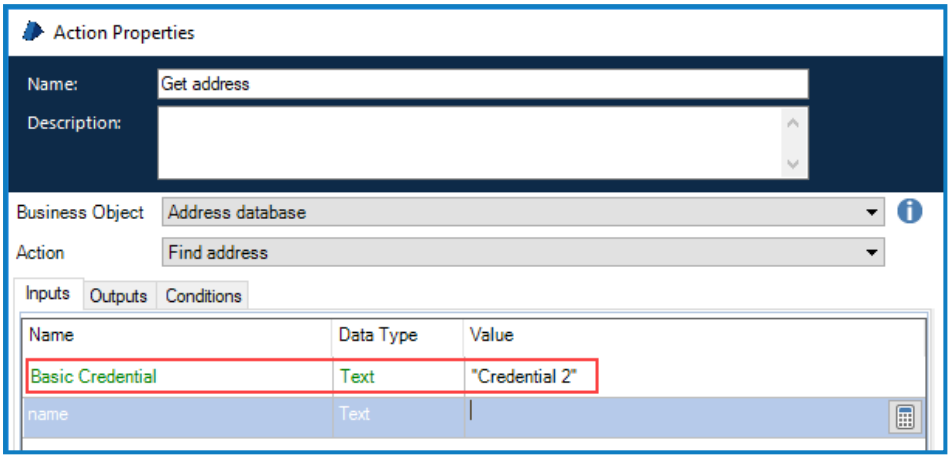


Credentials are used to store the data required to make an authenticated request to a Web API. Credentials applicable to the above authentication types are created in Credential Manager and then used to authenticate any request to the Web API. The credential selected in the API definition is used for all associated actions. However, when exposed to processes, an alternative credential can be used when specified as an input parameter.

In the example below, the *Address Credential* has been selected in the API definition as the default credential for all associated actions where an alternative value is not specified in the action inputs in Object Studio and Process Studio.



Given that **Expose to process** has been selected, a related input parameter is available for processes. The name of an alternative credential has been specified in the input parameter and is used when the API is called.



Authentication types

The following authentication types can be configured within a Web API definition.:

None

The API does not require authentication.

Basic

Uses HTTP Basic Authentication to send a username and password from a selected credential in request headers. If required, and if the API supports it, select the **Pre-emtively send the authorization header** check box. Pre-emptive authorization sends the basic authentication response directly with the HTTP request rather than when responding to an unauthorized response.

Select the required credential from the drop-down list and if required, select **Expose to process** and enter a parameter name.

OAuth 2.0 (Client Credentials)

Makes a request to an authorization server with a client ID and secret. If the credentials are valid, an access token is returned which is used to authenticate the API request.

The authentication type is configured using the following fields:

- **Authorization URI** – The address of the authorization server.
- **Credential** – The name of the credential, in Credential Manager, that contains the Client ID and Client Secret that is used to get the access token from the authorization server
- **Parameter name** – If the authentication is exposed to objects and processes, the name of the input parameter for actions.
- **Scope** – The level of access requested from the authorization server that will be permitted by the token.

OAuth 2.0 (JWT Bearer Token)

This credential type is a form of OAuth 2.0 authentication using a JSON Web Token (JWT) to authenticate rather providing a client ID and secret as with standard OAuth 2.0. The request data, specifying who is requesting the information and their intended use, is sent to the authorization server as a signed JWT. If the JWT is valid, the server returns an access token that is used to authenticate the API request.

The following fields are used to create the request.

- **Algorithm** – The algorithm used to sign the JWT – currently only the RSA SHA-256 algorithm type is supported.
- **Authorization URI** – The address of the authorization server.
- **Audience** – Used to identify the authorization server as the intended recipient of the token.
- **Scope** – The level of access requested from the authorization server to be permitted by the token.
- **Subject** – The Subject typically identifies the user for which the access token is being requested. Parameters can be used in this field.
- **JWT Expiry** – The length of time after which the token is not accepted by the authorization server.
- **Credential** – The credential, configured in Credential Manager, used to authenticate the request.
- **Parameter Name** – If the authentication is exposed to objects and processes, the name of the input parameter for actions.

Bearer Token

This credential type caters for situations where the token has already been obtained and needs to be sent in the authorization header. This could be used to support situations whereby Blue Prism uses a different method, such as a code stage, to obtain the token.

The token is stored in a credential and referenced in an API definition. When used in request headers the token passed in the following format:

```
Authorization: Bearer <AccessToken>
```

For example, Authorization: `Bearer FGRS5-PUUDW-NBC2Q-96UYR-QBDSY`

Custom

Custom credentials allow bespoke authentication methods to be used in Blue Prism. For example, a custom credential could be used to authenticate using an API key or subscription key, stored securely in the password field of a credential. The credential, stored in Credential Manager can be passed into the API request as a parameter allowing it to be used in the request body, header, or URL.

For more information about using custom credentials, see Custom credentials.

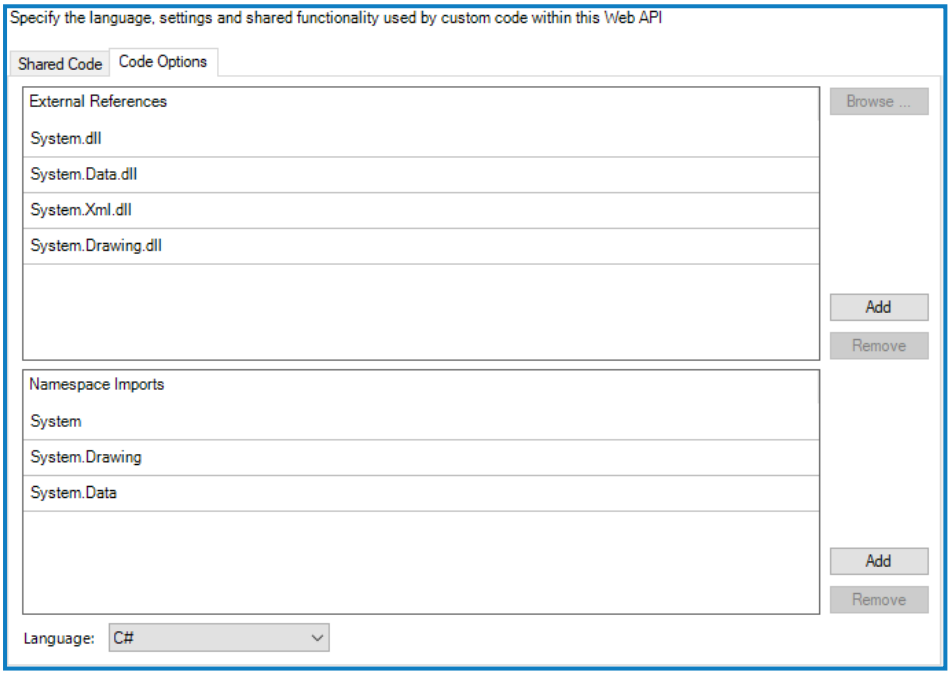
Common code

Some third-party APIs require more complex request and response data. The Share Code and Code Options settings are used when configuring code in individual actions to generate the body content for a request or to transform the HTTP response into output parameters.

Code options

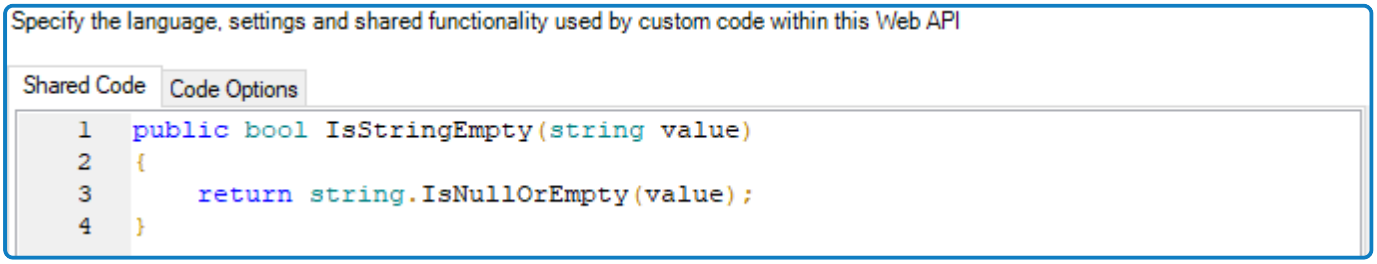
Specify the language to use for all code in the API definition and add any third-party libraries and the related Namespace Imports. The following languages are supported:

- Visual Basic
- C#



Shared code

Add any code, such as library functions and common properties, that is required across the whole API to define HTTP body content and responses.

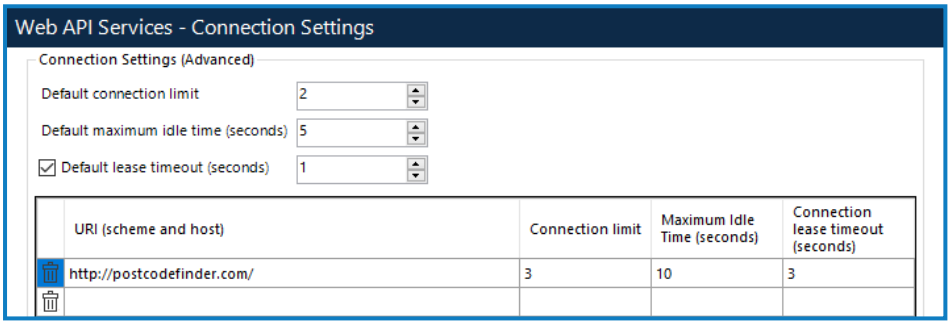


Connection settings

Configure advanced connection settings, applied when making web requests. Any user with the System – Web Connection Settings permission enabled for their user role can configure these settings.

Default settings can be configured for all Web API requests. The default settings are overridden if values are set for specific URIs:

- **URI** – The URI for which the settings will be applied. The URI must only be comprised of scheme and host and the settings are applied to all requests that use the entered URI.
- **Connection limit** – The number of connections per service point.
- **Connection lease timeout** – The time permitted for a connection to remain idle before it closes.
- **Maximum idle time** – The number of seconds after which a connection is closed.



Web API credentials

For API requests that require authorization, it is necessary to add the relevant credentials to Credential Manager.

1. Select **System** > **Security** > **Credentials** and click **New**.

   The Credentials Details dialog displays.

2. Select the authentication type for the credential:

   - **General** – Add the authentication username and password required to create credentials for non Web-API authentication.
   - **Basic** – Add the authentication username and password required to create the basic authentication header.
   - **Bearer Token** – Add the Bearer token to add to the authentication header.
   - **OAuth 2.0 (Client Credentials)** – Add the client identifier and secret, stored in a credential, to retrieve an OAuth 2.0 access token from an authorization server. The access token is then used to authenticate Web API requests.
   - **OAuth 2.0 (JWT Bearer Token)** – Add the JWT issuer and private key, used to sign the JWT – a PKCS8 private key is expected in a PEM base64-encoded format. The JWT is then sent to the authorization server to get the access token required to authenticate the request.

   > If the selected authentication type requires a username and password, HTTP password authentication protocol dictates that passwords can only contain any of the first 128 ASCII characters. If any other characters are used, Blue Prism replaces the character with ? resulting in a failed login attempt.

   For further information about configuring authentication types in API definitions, see Common authentication.

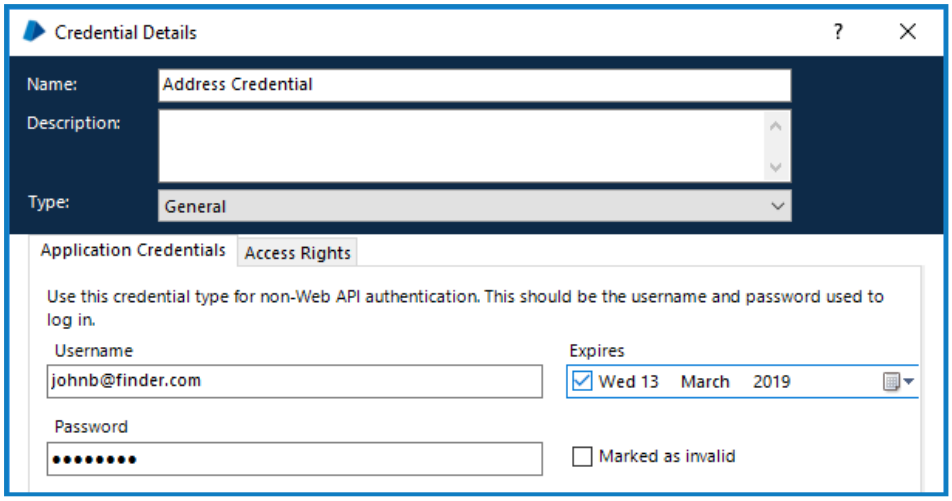
3. In the Additional Properties section, add any general purpose, named properties for the credential. The values are held securely within the database and can be requested by processes.

4. Select the **Access Rights** tab and define the required permissions.

   It is recommended that the access rights are configured so only user accounts with the appropriate security roles are able to access to each credential. When a credential is limited by security role:

   - The selected security roles must be held by the user(s) who require access to the credential when building or debugging a process or object.
   - Only runtime resources that are configured to explicitly authenticate will potentially be able to access the credential.
   - The selected security roles must be held by the account(s) used by runtime resources to authenticate against the environment.

5. Click **OK** to save the credential.

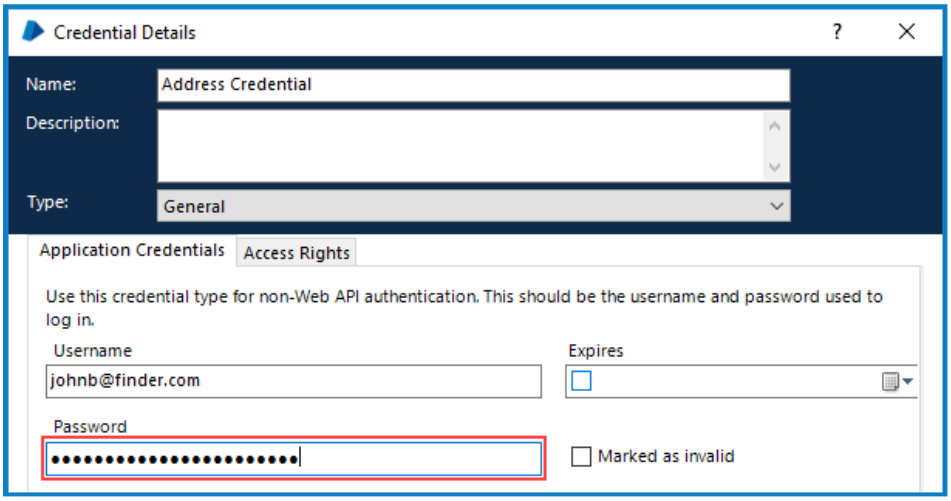


Web API custom credentials

This example demonstrates how to pass an API key, securely stored in a credential, into a request URL as a query string.

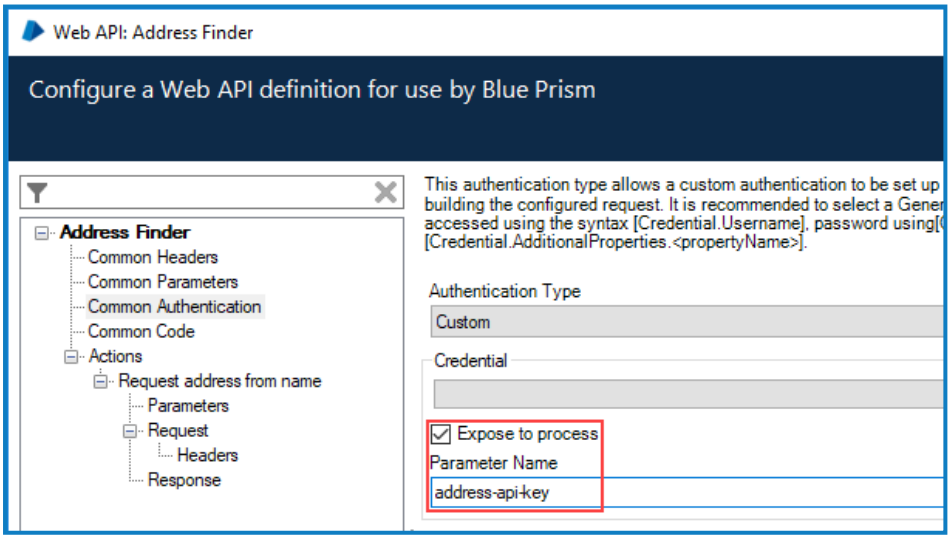
Configure the credential

An API key is stored in the password field of a credential.

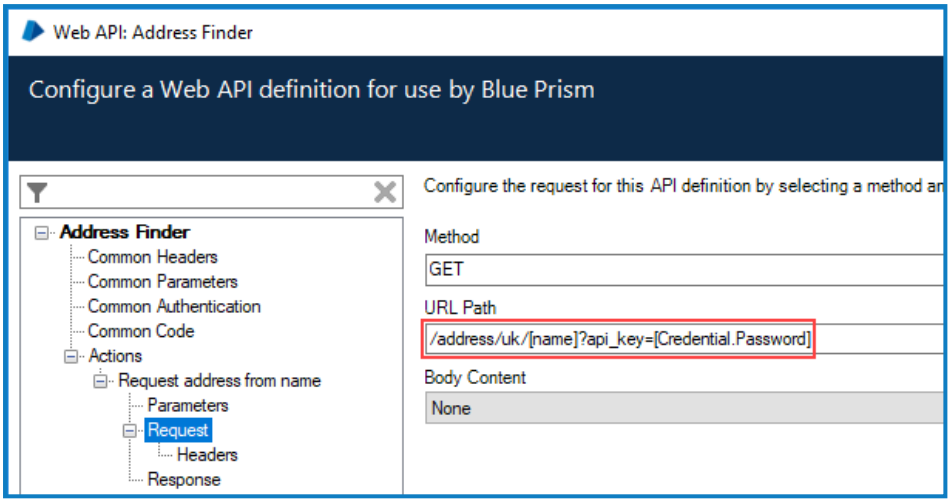


Configure the Web API definition

The Custom authentication type is selected in the Common Authentication screen for an API definition. A credential does not need to be selected as it is not being passed in the request header. Instead, **Expose to process** is selected to create an input parameter that will determine the required credential in a process.



In the URL for the request, enter **[Credential.Password]** in place of the API key.

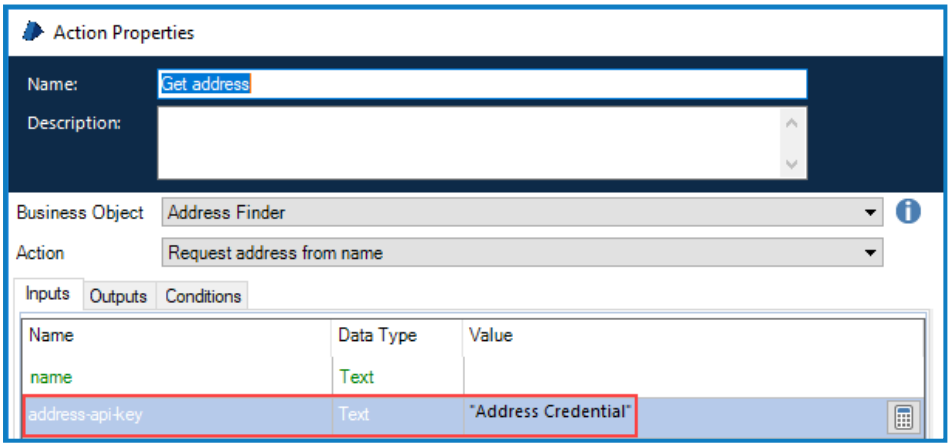


This parametrizes the password field of the given credential, allowing the value to be passed. The same syntax can be used to pass other values from a credential:

- [Credential.Password]
- [Credential.UserName]
- [Credential.AdditionalProperties<propertyname>]

Specify the credential in an object or process

The required business object and action are selected in a process action and the parameter set in the API definition is available as an input parameter. The value of the parameter is the name of the required credential in quotation marks.



When the object or process is run, the API key from the credential is used in the request:

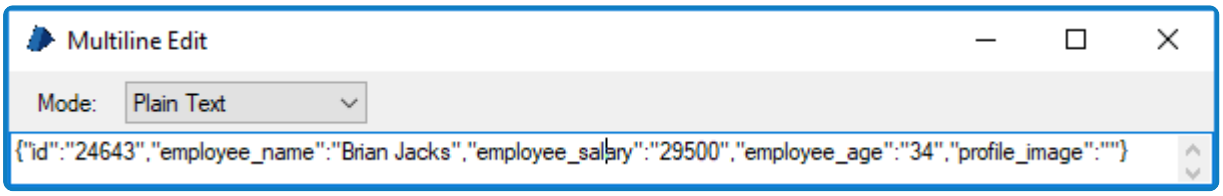
```
GET http://www.postcodeuk.com/address/UK/"Andy Taylor"?api_key=FGRS5-PUUDW-NBC2Q-96UYR-QBDSY
```

Extract response data with JSONPath

When a response body content is JSON, JSONPath can be used to extract specific elements of the JSON into output parameters. The output parameters have a specified Blue Prism data type, and the object found using the JSONPath is converted to that data type. This example demonstrates how to extract data from a JSON HTTP response, storing it in stages in a process.

Configure the API definition response

An API definition returns the following JSON response:

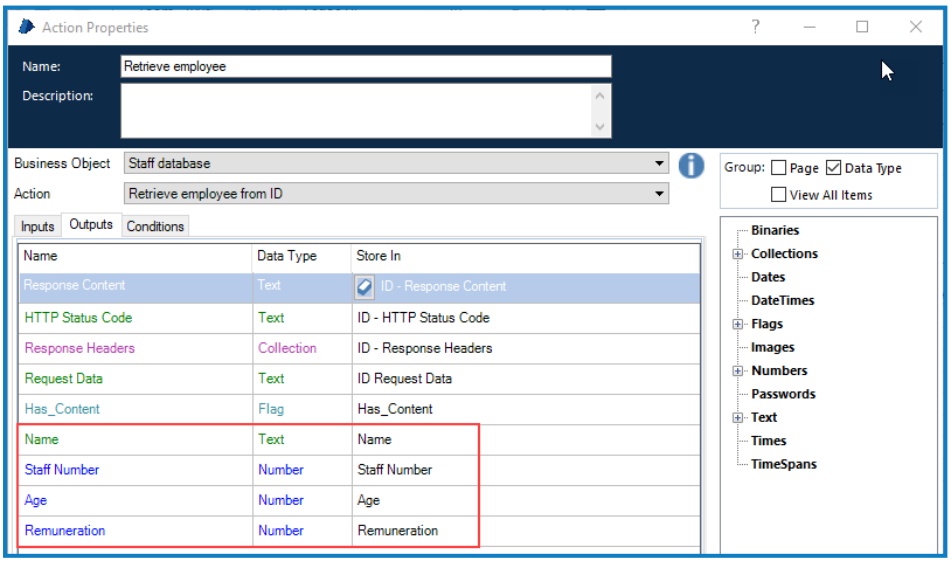


To extract this data, select the Response for the required action in the API definition. Add a parameter for each data item selecting the appropriate data type and method.

	Parameter	Data Type	Method	Json Path
	Remuneration	Number	Json Path	$.employee_salary
	Staff Number	Number	Json Path	$.id
	Name	Text	Json Path	$.employee_name
	Age	Number	Json Path	$.employee_age

View the response in Process Studio

In Process Studio, select the Outputs for the process that uses the Web API action for which the response was configured. Outputs have been added corresponding to the parameter name added to the response and the related data items have been created in the process to store each parameter.



When the process is run, data from the response is stored in the appropriate stage allowing it to be easily used elsewhere in the process.

If the JSONPath defined for the output parameter does not find a match in an HTTP response, the output parameter will return an empty data item/collection. If the JSONPath defined for the output parameter finds multiple possible matches in an HTTP response, an exception will be thrown in the process.

JSONPath syntax

Blue Prism supports the following syntax for JSONPath expressions:

JSONPath	Description
$	The root object or element.
@	The current object or element.
. or []	Child operator.
..	Recursive descent.
*	Wildcard.
[]	The native array operator.
[start:end:step]	Array slice operator.
?()	Applies a filter (script) expression.
()	Script expression, using the underlying script engine.

Example

A staff database contains a number of records that includes data about the employees of an organization.

```
{
"staff-database": {
"employee": [
{
"id": 1245,
"name": "Geoff Bryant",
"salary": 45000,
"age": 46
},
{
"id": 1365,
"name": "Tom Roberts",
"salary": 38000,
"job-title": "developer"
"age": 34
},
{
"id": 1287,
"name": "Herman Blunt",
"salary": 27000,
"job-title": "tester"
"age": 26
},
]
}
}
```

The following are examples of how JSONPath can be used to extract data.

JSONPath	Result
$.staff-database..salary	Returns the salary for all employee records.
$.staff-database['employee']	Returns all employee records.
$..employee[?(@.job-title)]	Filters all records with a job title.
$..employee[?(@.salary<40000)]	Filters all records that have a salary of less than 40000.

Sending files

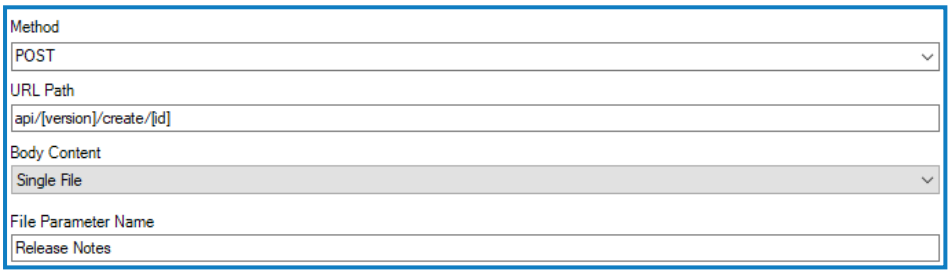
The following examples demonstrate how to send single or multiple files as part of an HTTP request.

Send a single file

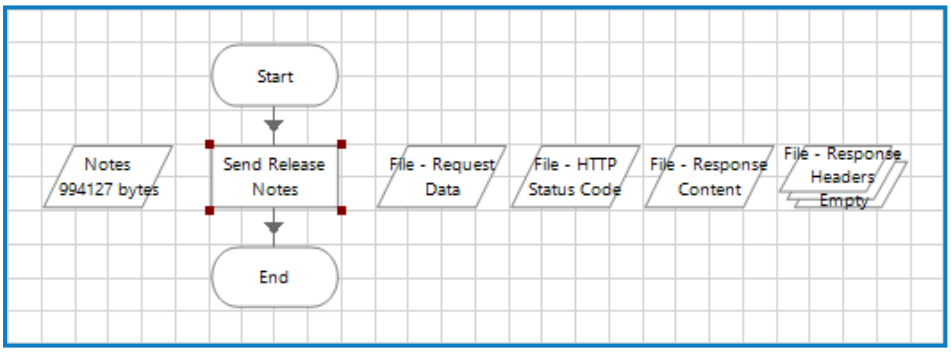
This sends a single file as a binary stream. The Content-Type HTTP Header is set to application/octet-stream when using this content type unless a different Content-Type header value has been specified in the Web API action definition.

Web API definition

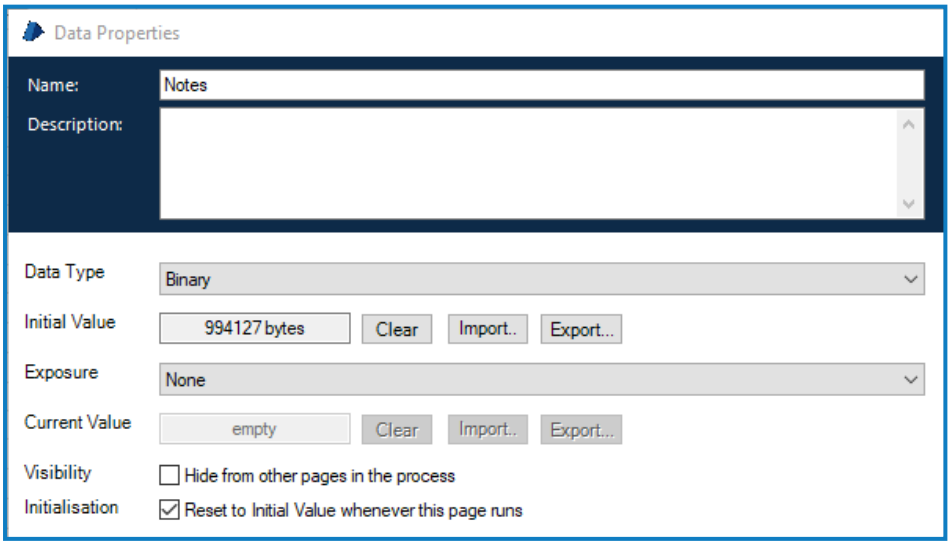
An action in an API definition uses the **POST** method and **Single File** for the body content. The File Parameter Name creates an input parameter named *Release Notes* in processes that use this action.



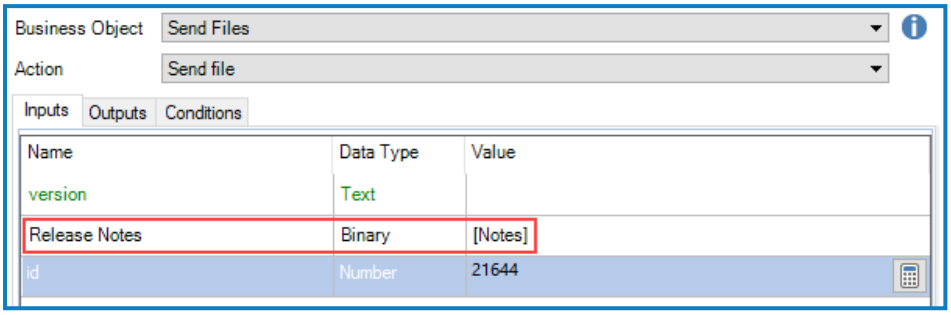
Process



In Process Studio, a process contains a data item called Notes with a Data Type of Binary into which a file has been imported.



An action stage in the process uses the API definition and references the Notes data item as the value for the input parameter.



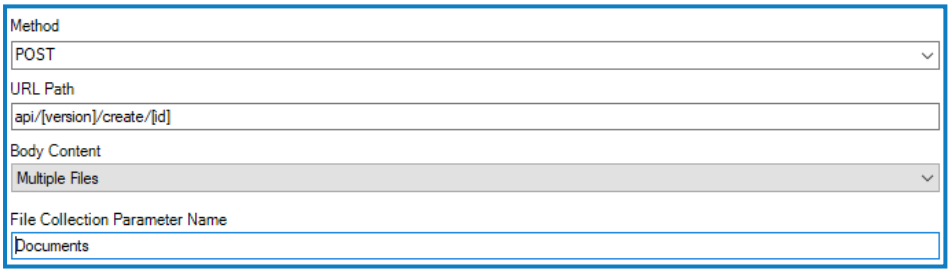
When the API is called, the file imported into the data item is sent as part of the HTTP request.

Send multiple files

Multiple files are sent in a single request using the multipart/form-data content-type.

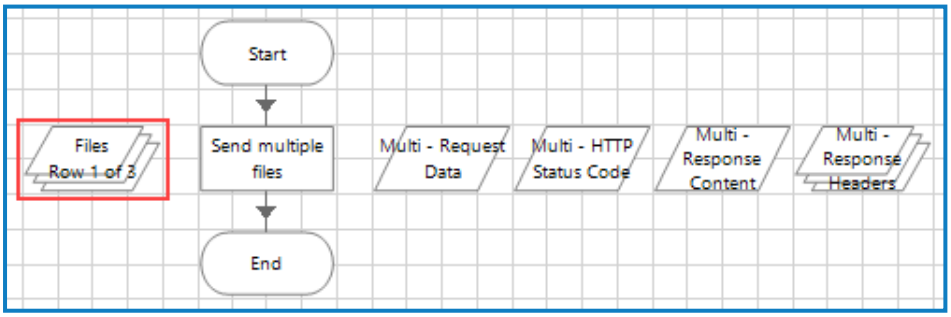
Web API definition

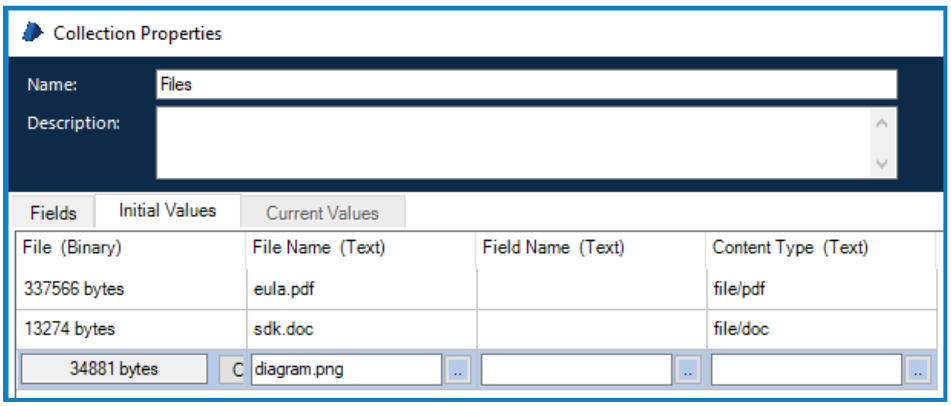
An action in an API definition uses the **POST** method and **Multiple Files** for the body content. The File Parameter Name creates an input parameter named *Documents* in processes that use this action.



Process

A process contains a collection item called *Files* into which a number of files have been imported.



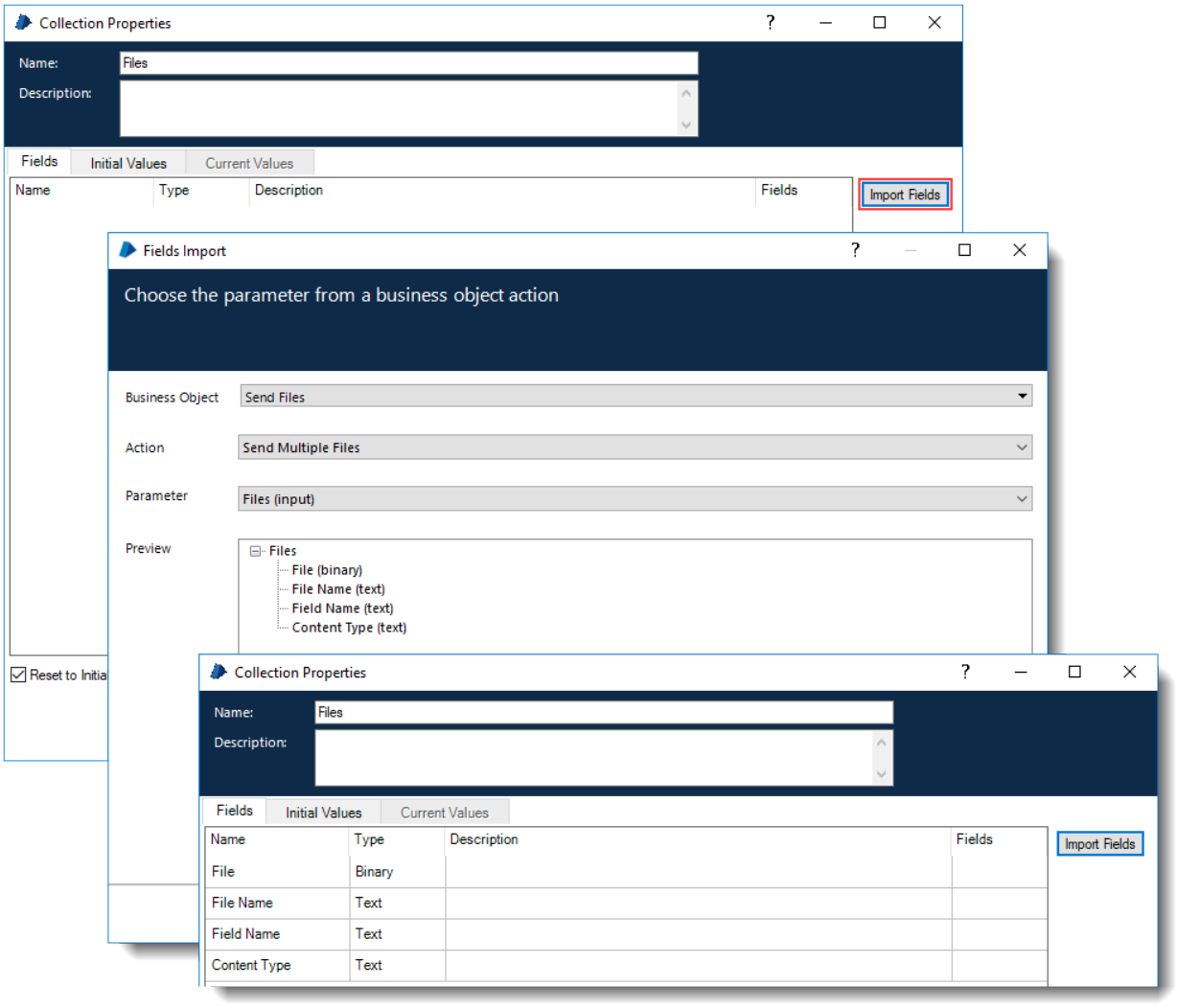


The third-party Web API may require the File Name, Field Name, and Content Type fields to be populated. Please refer to the third-party Web API documentation.

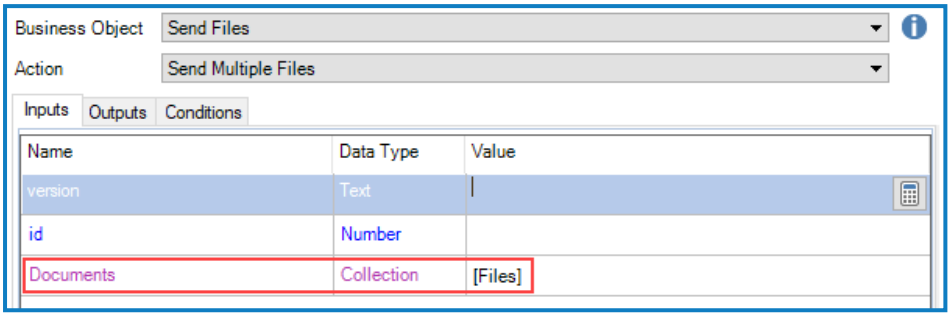
Collections passed into input parameters are expected to use the following specified schema:

Name	Data Type	Is Required	Description
File	Binary	True	The file to be sent.
File Name	Text	False	Used to populate the filename header for this file's section in the multi-part form body
Field Name	Text	False	Used to populate the name header for this file's section in the multi-part form body
Content Type	Text	False	Used to populate the Content-Type header for this file's section in the multi-part form body. If a value is not specified, *application/octet-stream* is used by default.

When creating a collection, the required schema can be imported by clicking **Import Fields** from the Collection Properties and selecting a parameter from required web API action.



An action stage in the process uses the API definition and references the *Files* data item as the value for the input parameter.



When the API is called, the file is sent as part of the HTTP request.