Adroit Technologies

Adroit Edge Gateway

HTTP Microservice

HTTP

Adroit
Edge Gateway

COPYRIGHT

Advanced Worx 112 (Pty) Ltd t/a Adroit Technologies
20 Waterford Office Park
189 Witkoppen Road (c/o Waterford Drive)
Fourways
South Africa

www.adroitscada.com

Version	Date	Author	Comment	Applicable Edge Version	Approval
1.0	06.05.2025	D. Acuna	Initial Draft	10.0.5.5	
2.0	18.05.2025	D. Acuna	Final Draft	10.0.5.5	
3.0	19.08.2025	D. Wibberley	Final	10.0.5.5	

Contents

1. Introduction

The HTTP Publisher is a Microservice that enables the Adroit Edge Gateway to publish operational data to a HTTP URL endpoint. This data can then be used in Dashboards, Reports and Analytics using the AI capabilities of the chosen platform.

2. Configuration and Usage

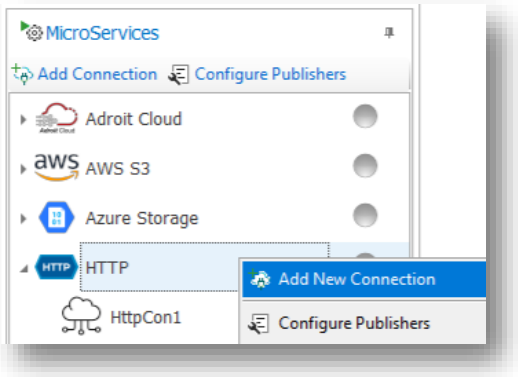
Configuration of all Adroit Edge Publishers is done within the **Config Editor**, under: **Configuration** → **Welcome** → **Edge Gateway Configuration**.

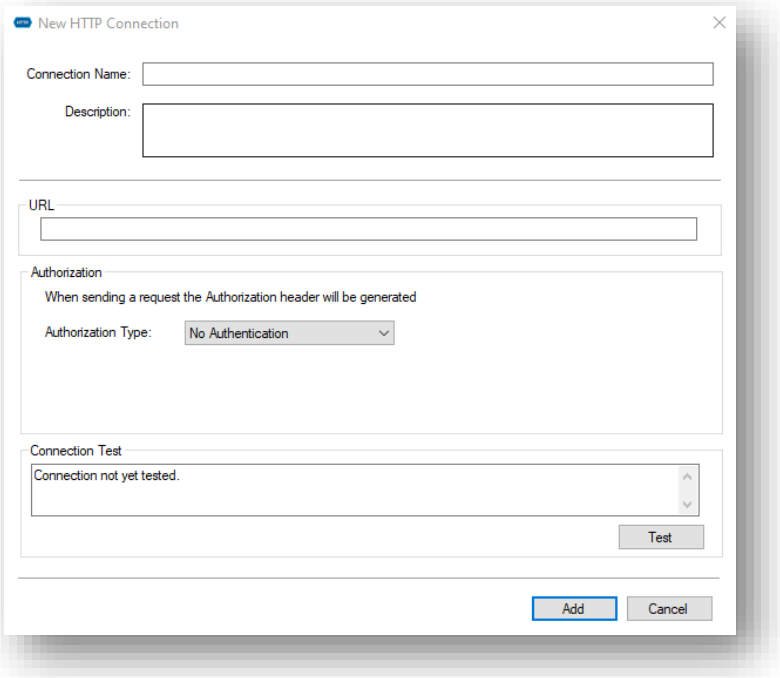
2.1. HTTP Connections

2.1.1. Adding a New Connection

To create a new HTTP connection:

1. Right-click the **HTTP** Microservice in the treeview.
2. Select the option to create a new connection.





In the **Add HTTP Connection** dialog, configure the following settings:

- **Connection Name**: The display name of the connection under the HTTP Microservice.
- **Description**: A description of the connection. Used as a brief tooltip description for the connection.
- **URL**: The HTTP URL endpoint.
- **Authorization**: Different types of authorization can be used for the HTTP request. When sending a HTTP request the Authorization header will be generated for you for the various authorization types below:
  - **No Authentication:** In this case, no credentials are required. The request is made openly without any verification process. This is typically used for public APIs or endpoints that don't restrict access.
  - **Basic Authentication:** This method requires a username and password to access the resource. The credentials are sent in the request header, encoded in Base64.

    Example of a Basic Authentication header:
    Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=
    Note: *This method is not secure unless used over HTTPS, as the credentials can be decoded easily.*

  - **Bearer Token:** This method uses a token instead of a username and password. The token is typically generated by an authentication service and is included in the request header.

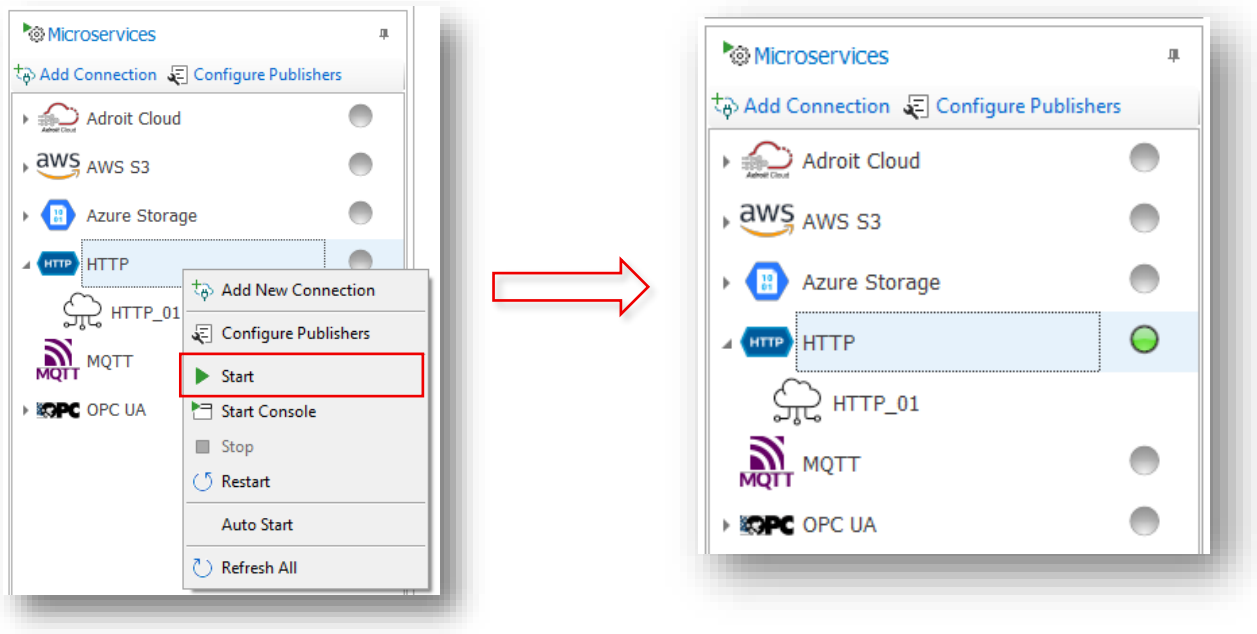
    Example of a bearer token header:
    Authorization: Bearer abc123xyz
    Note: *This method is commonly used in APIs that follow OAuth 2.0 authentication*

Each authentication method serves a different purpose depending on the security needs of the application

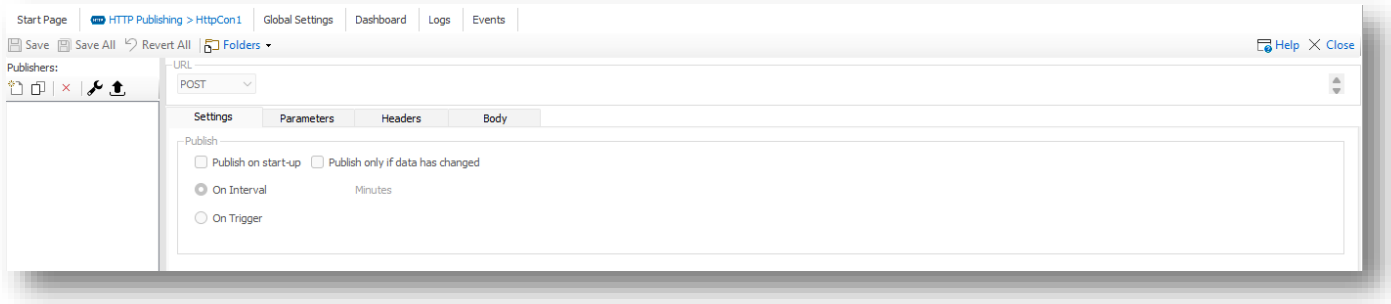
Connection Testing
A connection test feature is available to verify the URL and endpoint configuration.

Edit or Delete a Connection
To edit or delete a connection, simply right-click on the connection and select the required option.
Note: *Deleting a connection will delete any associated Publishers configured for the connection.*

Start the Connection
To start he Microservice, right-click on the required Microservice and select Start.



After creating a connection, double-clicking it will present the configuration options for **HTTP Publishers**. You can add multiple Publishers to a connection and configure each one with these two tabs:



These tabs are:

Tab	Description
Settings	General settings, including HTTP request method i.e. POST, DELETE etc. and publishing modes (e.g., interval or trigger-based).
Parameters	These define the data that is sent within the body of the HTTP request.
Headers	These provide additional information about the HTTP request, such as authentication tokens, content type, and API-specific settings
Body	Configure Dataelement substitutions and body message for the HTTP URL endpoint. Validate and preview the final body message with live data and test POST the body message to the HTTP URL endpoint.

2.2. HTTP Publishers

After creating a connection, multiple Publishers can be added by selecting the connection in the treeview and clicking **Configure Publishers** from the toolbar or by double-clicking it.

2.2.1. Adding Publishers

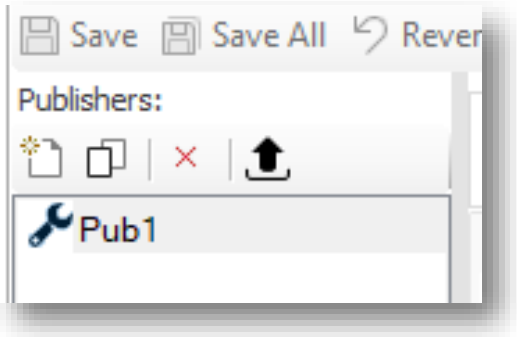
To add a Publisher:

1. Click the **Add New Publisher** from the toolbar.
2. Provide a name and description for the Publisher.

Each Publisher starts in **Maintenance Mode** (not actively publishing) by default.
To switch to **Publish Mode**, use the context menu or toolbar buttons options.

Additional options allow you to:

- Edit the Publisher's description.
- Delete the Publisher.
- Copy the Publisher, including all settings, substitutions, and body configurations. The copy function also supports Find and Replace for substitution Dataelement names.

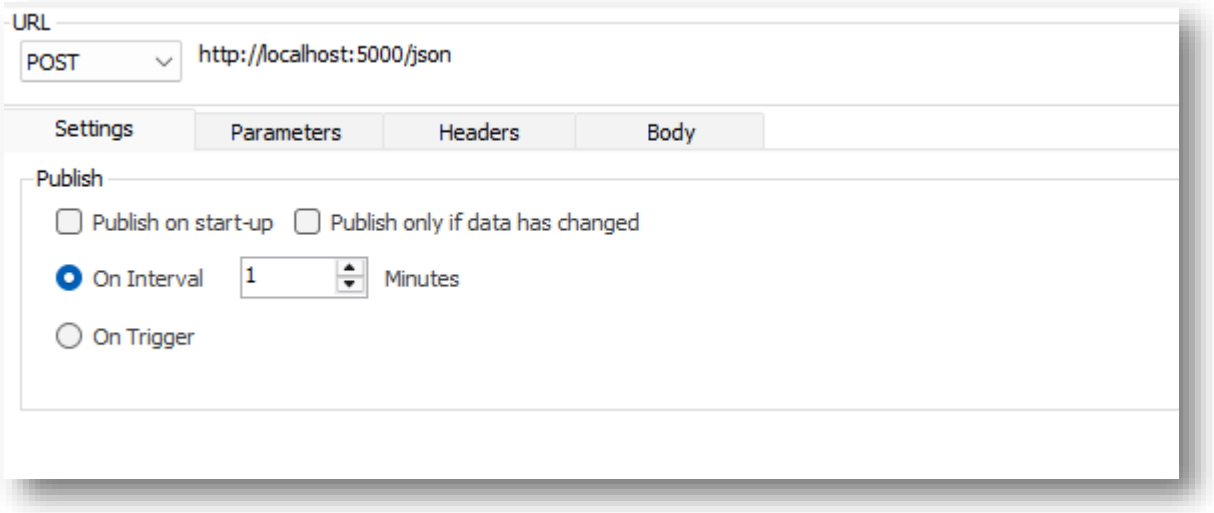


2.2.2. Publisher Settings

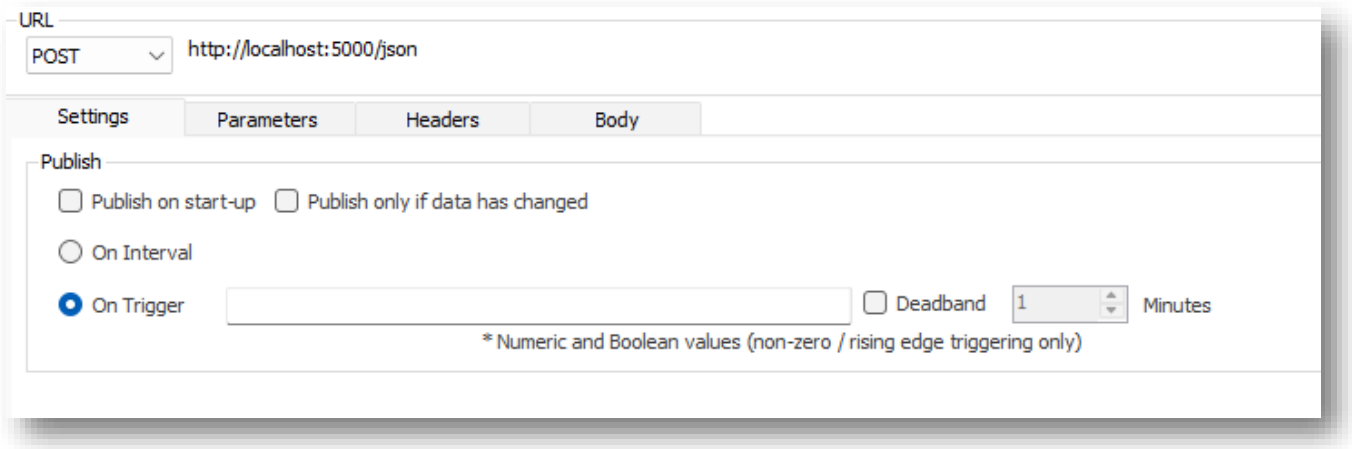
The **Settings** tab includes:

- **HTTP Request Method**: This is the HTTP request method to be used to send data to the HTTP URL endpoint i.e. POST, DELETE, PUT and PATCH.
  - **POST**: Used to send data to the server to create a new resource. Adds a database entry.
  - **DELETE**: As the name suggests, this request is used to remove a resource from the server. It's commonly used in RESTful APIs to delete database entries.
  - **PUT**: Used to update or create a resource. Unlike POST, PUT typically replaces the entire resource, whereas POST adds a new one. Can be used to update a current database entry.
  - **PATCH**: Similar to PUT, but it applies partial updates to a resource rather than replacing it entirely. Can be used to update a part of a database entry.

- Publish Mode Settings:
  - **On Interval**: Publishes messages at a regular interval (defined in minutes).
    - **Interval** (in minutes): Specifies the publishing interval.



  - **On Trigger**: Publishes messages triggered by a Dataelement value change, provided that the trigger Dataelement has changed and holds a non-zero value.
    - **Deadband**: Defines the waiting period (in minutes) before accepting the next trigger after the last published time.



For both publishing modes above you can also:

- **Publish on start-up:** This publishes on startup of the microservice and when the Publisher is set to publish mode.
- **Publish only if data has changed**: Publishes only if any substitution Dataelement values have changed.

2.2.3. Parameters Settings

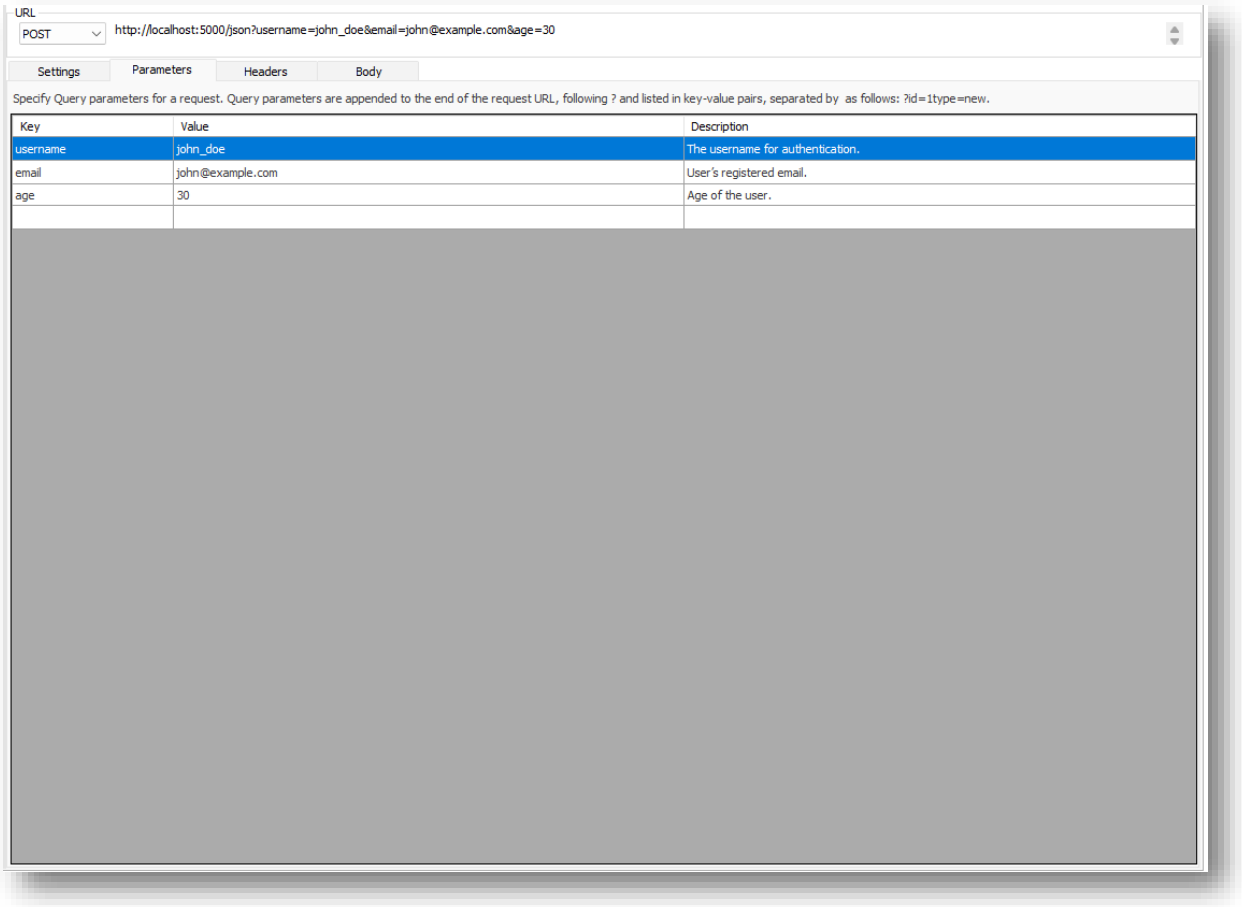
When making an HTTP POST request, both parameters and headers play a crucial role in defining the request's behavior. These settings are typically structured in a grid format consisting of three columns: Key, Value, and Description. This format helps ensure clarity and ease of configuration.

- The **parameters** define the data that is sent within the body of the HTTP request.
- They are structured in a grid with the following columns:
  - **Key**: The name of the parameter.
  - **Value**: The actual data being sent.
  - **Description**: A brief explanation of the parameter's purpose.
- These parameters help build the URL and determine how the server processes the request.

Example:

Key	Value	Description
username	john_doe	The username for authentication.
email	john@example.com	User's registered email.
age	30	Age of the user.

By structuring parameters and headers in this manner, users can efficiently configure HTTP requests, ensuring clear communication with the server.



2.2.4. Headers Settings

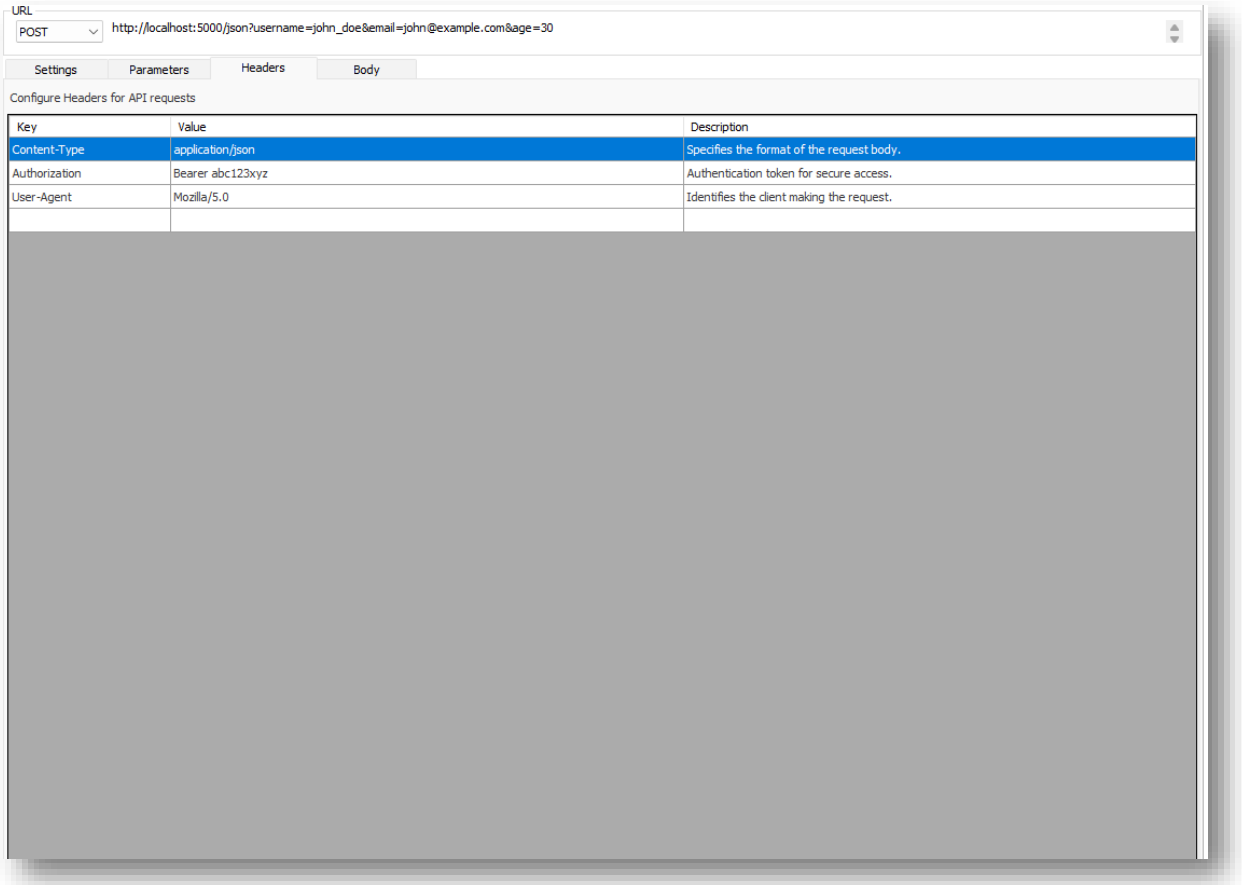
When making an HTTP POST request, both parameters and headers play a crucial role in defining the request's behavior. These settings are typically structured in a grid format consisting of three columns: Key, Value, and Description. This format helps ensure clarity and ease of configuration.

- **Headers** provide additional information about the HTTP request, such as authentication tokens, content type, and API-specific settings.
- They follow the same grid format:
  - **Key**: The name of the header.
  - **Value**: The associated value.
  - **Description**: Explanation of the header's role in the request.

Example:

Key	Value	Description
Content-Type	application/json	Specifies the format of the request body.
Authorization	Bearer abc123xyz	Authentication token for secure access.
User-Agent	Mozilla/5.0	Identifies the client making the request.

By structuring parameters and headers in this manner, users can efficiently configure HTTP requests, ensuring clear communication with the server.



2.2.5. Body

In the Body tab we can configure the substitution Dataelements for the HTTP Publisher as well as the body messages which will be sent.
We are also provided with an option to validate, preview and test the substituted body message before it's been sent.

2.2.5.1. Substitutions

The Substitutions grid allows users to define multiple substitutions with corresponding Dataelements and formatting.
To add Dataelements to the Substitutions grid:

1. Navigate to the Dataelements tree view under the relevant Adroit Datasource.
2. Select the desired Dataelements (you can multiselect) and drag them into the Substitutions grid.
3. Alternatively, you can directly input the full names of the Dataelements into the Substitutions grid.

Additional Substitutions functionalities include:

- **CSV Import/Export**: Handle substitution lists and their configurations using CSV files. Import or export them seamlessly within the Substitution Grid.
- **Find and Replace**: Locate and modify text within substitution Dataelement names directly in the Substitution Grid.

2.2.5.2. Body

The Body configuration provides a markup editor for creating the JSON body. Substitutions from the grid can be inserted into the JSON by selecting a substitution and using the Body toolbar buttons to insert a substitution of specific Dataelement's data i.e. its value, quality and timestamp:

- **Value**: Inserts a ${substitution name}, which will substitute in the Dataelements value.
- **Quality**: Inserts a ${substitution name}ST, which will substitute in the Dataelements quality i.e. bad, good etc.
- **Timestamp**: Inserts a ${substitution name}TS, which will substitute in the Dataelements timestamp

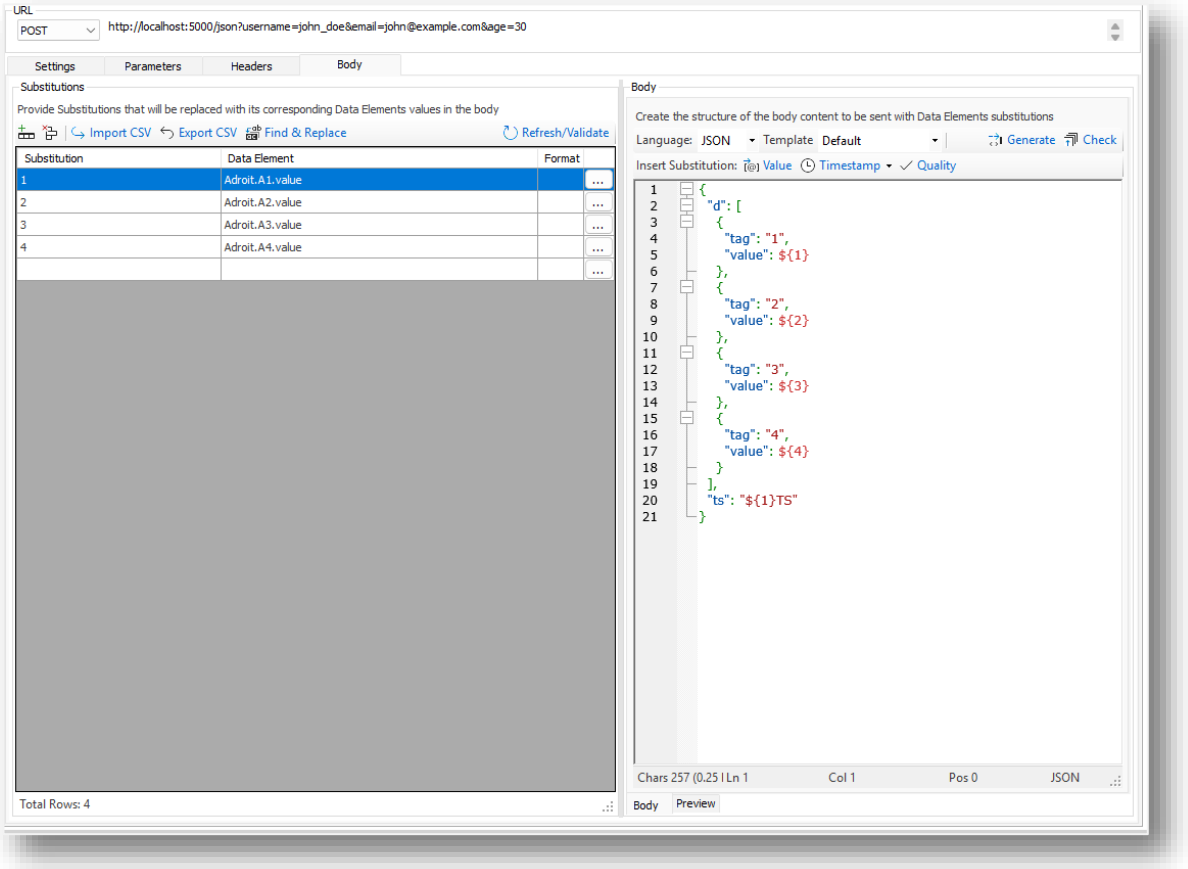
These Substitutions can be also added manually into the JSON in the markup editor.
Additional Body features include:

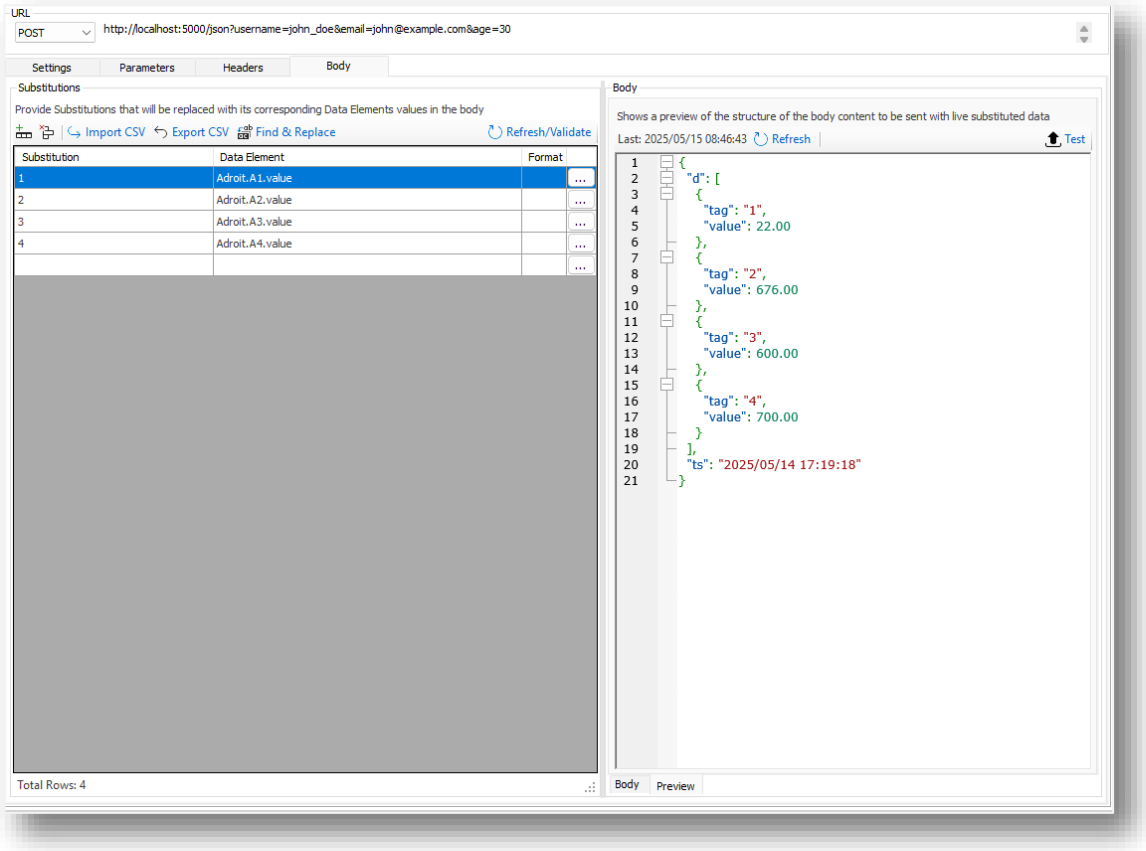
- **Templates**: Create JSON bodys using predefined imported templates. A default template is available to generate a JSON structure containing key-value pairs for each substitution, such as *substitution name: ${substitution name}*
- **Validation**: Verify the JSON body to ensure accuracy and receive feedback on any errors.

2.2.5.3. Preview and Test

The **Preview** tab allows you to:

- **View** – View the final JSON body with live substituted values.
- **Test**-Test post the body message to ensure functionality. If the test fails, an error response will be displayed.


URL
POST
http://localhost:5000/json?username=john_doe&email=john@example.com&age=30
Settings
Parameters
Headers
Body
Substitutions
Provide Substitutions that will be replaced with its corresponding Data Elements values in the body
Import CSV
Export CSV
Find & Replace
Refresh/Validate
Substitution
Data Element
Format
1
Adroit.A1.value
2
Adroit.A2.value
3
Adroit.A3.value
4
Adroit.A4.value
Total Rows: 4
Body
Create the structure of the body content to be sent with Data Elements substitutions
Language: JSON
Template Default
Generate
Check
Insert Substitution: Value Timestamp Quality
{
"d": [
{
"tag": "1",
"value": ${1}
},
{
"tag": "2",
"value": ${2}
},
{
"tag": "3",
"value": ${3}
},
{
"tag": "4",
"value": ${4}
}
],
"ts": "${1}TS"
}
Chars 257 (0.25 I Ln 1
Col 1
Pos 0
JSON
Body
Preview


URL
POST
http://localhost:5000/json?username=john_doe&email=john@example.com&age=30
Settings
Parameters
Headers
Body
Substitutions
Provide Substitutions that will be replaced with its corresponding Data Elements values in the body
Import CSV
Export CSV
Find & Replace
Refresh/Validate
Substitution
Data Element
Format
1
Adroit.A1.value
2
Adroit.A2.value
3
Adroit.A3.value
4
Adroit.A4.value
Total Rows: 4
Body
Shows a preview of the structure of the body content to be sent with live substituted data
Last: 2025/05/15 08:46:43
Refresh
Test
{
"d": [
{
"tag": "1",
"value": 22.00
},
{
"tag": "2",
"value": 676.00
},
{
"tag": "3",
"value": 600.00
},
{
"tag": "4",
"value": 700.00
}
],
"ts": "2025/05/14 17:19:18"
}
Body
Preview

2.3. Work Folder Location

All configuration files related to the HTTP Publisher are stored in:
C:\ProgramData\Adroit Technologies\SmartDataServices\HTTP

Users typically do not need direct access to these folders, as configuration can be managed via the Config Editor. However, key files include:

- *config.xml* : Stores all settings, Dataelement substitutions, and body configurations.
- *HTTP.log* : Available only if "Log to file" is enabled. This log file must be opened manually.'

Subfolders include:

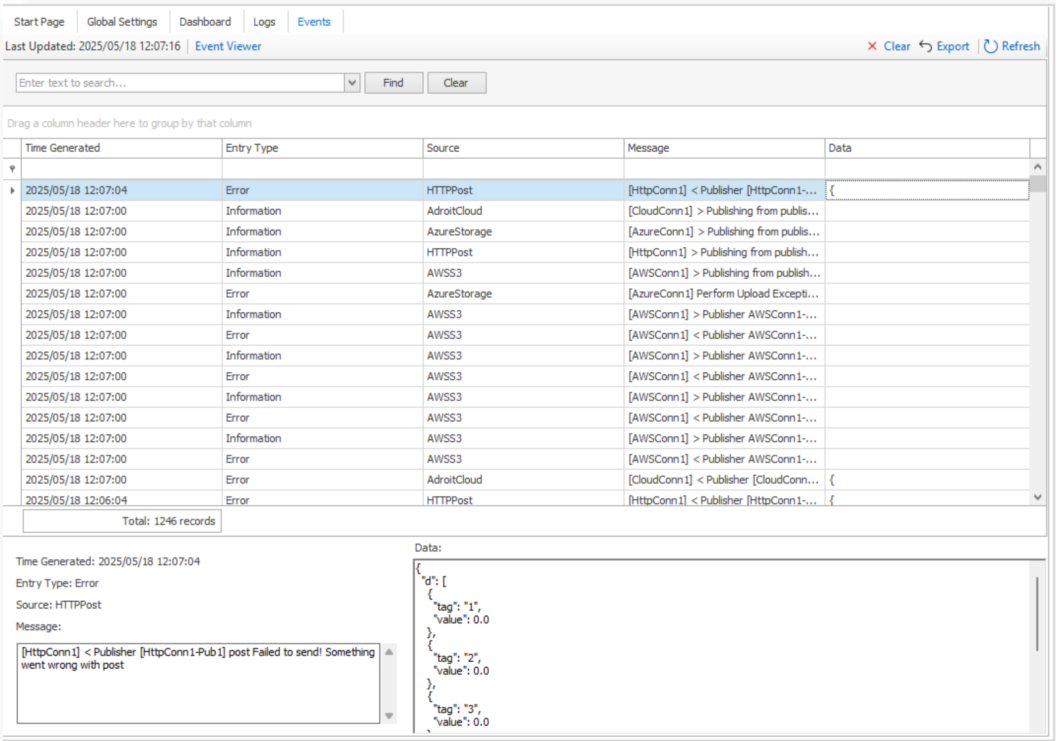
- CSV: Contains CSV files for importing and exporting Dataelement substitutions.
- Templates: Contains predefined JSON templates for generating specific body messages.

3. Troubleshooting

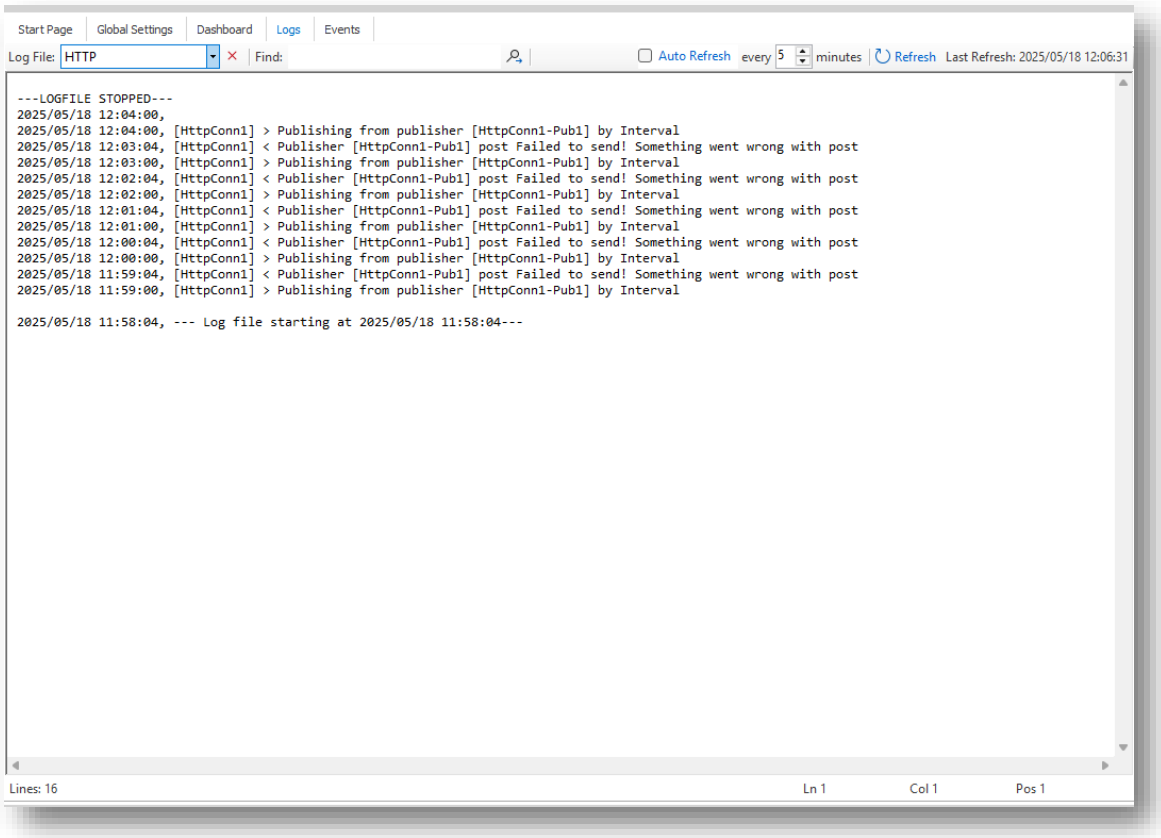
The HTTP Microservice provides various tools to help resolve issues:

- **Connection Testing**: The HTTP connection settings include a test functionality to verify the connection to the HTTP URL endpoint.
- **Substitution Validation**: In the HTTP Publishers configuration, the substitution grid highlights invalid Dataelements of a substitution in red for easy identification.
- **Body Validation**: The body configuration features a validation tool to ensure the JSON body is properly formatted, providing detailed feedback on any errors.
- **Preview and Testing**: The Preview tab allows users to preview the final JSON body with live substituted values and test-Post the body message.
- **Diagnostics**: Access additional tools on the main Adroit Edge Gateway tabs for advanced troubleshooting:

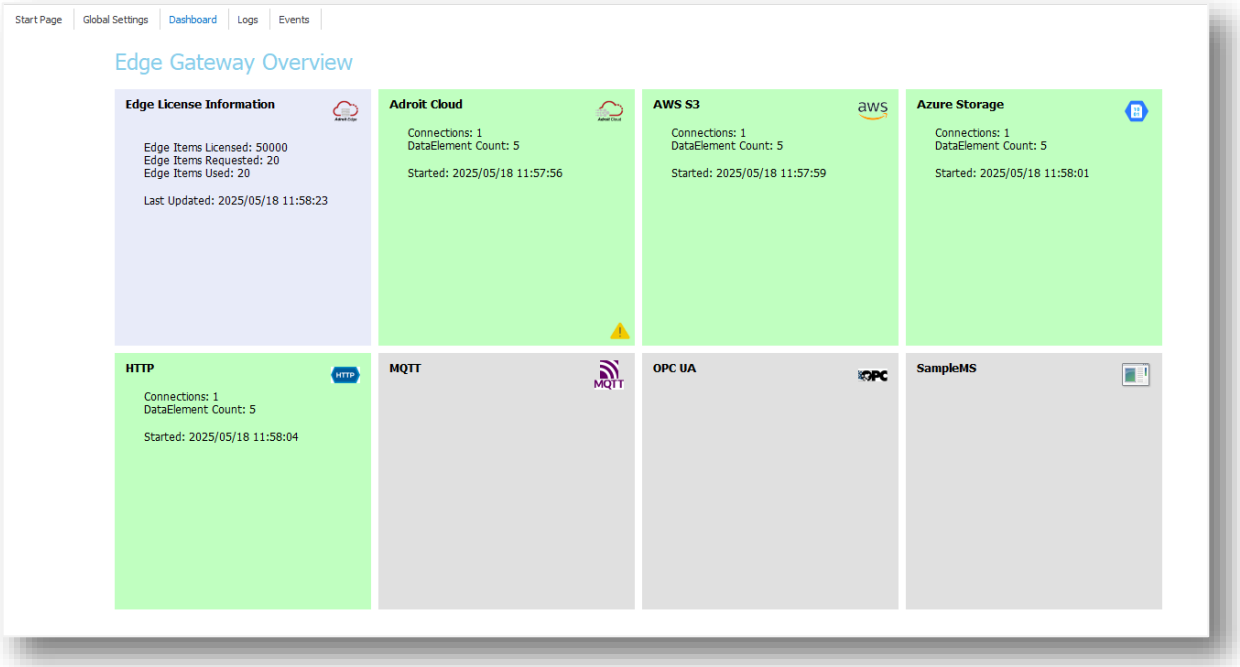
    - **Event Log Viewer**: View all Microservice event log messages sourced from the SmartDataServices event log.

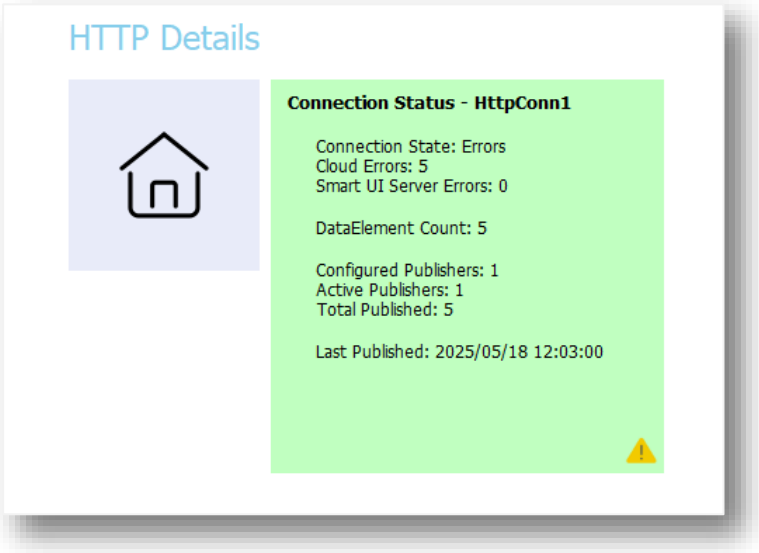


- **Log File Viewer**: Examine the complete contents of a selected Microservice log file.



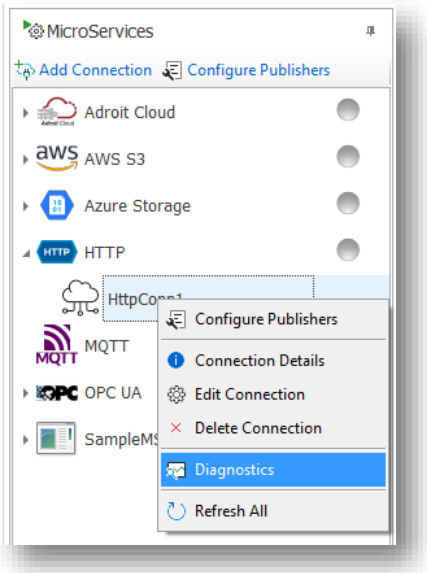
- **Dashboard**: Gain real-time dashboard-like feedback from Microservices, featuring specific Microservices statuses, counters, and indicators.

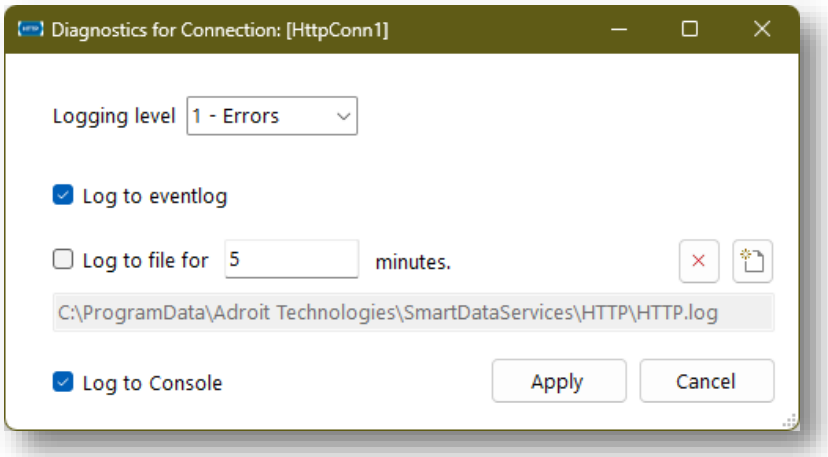




Diagnostics information logging, including its log level (i.e. Errors, Warnings, Information and Debug levels etc.) and output destination (i.e. Event log, log file, console etc.) is configured through the Diagnostics dialog for each HTTP connection.

To configure the Diagnostics for an HTTP connection, right-click the HTTP connection and click Diagnostics.





- **Microservice Console View**: Operate each Microservice as a console application to observe its real-time console messages. To run a Microservice as a console right- click the Microservice and click Start Console.