Ascent360 MEET YOUR DATA

Webhook Data Collection

In this article:

- The Basics
- Setup Information
- Request Body Parameters
- Retry Requirement

The Basics

What is a Webhook?

Webhooks offer a simple, real-time way to send data from client applications to Ascent360. They are widely used for efficient, event-driven data sharing.

Clients can integrate webhooks to send relevant data (e.g., orders or customer info) directly to Ascent360. Some development effort is required—please contact your Customer Success Manager or the Help Desk before starting.

Example: A POS or PMS system can send purchase data to Ascent360 via webhook. This data can:

- Be added to your database as an integration feed
- Trigger real-time actions, like a custom transaction confirmation email

Ascent360 receives, parses, and stores the data automatically.

How It Works

Ascent360 uses a simple, cloud-hosted application to receive data via HTTP POST at:
`https://webhook.ascent360.com/V1/Data/{GUID}`

Here's what happens:

1. The client sends a JSON or XML payload to the Ascent360 webhook URL.
2. The URL includes a client-specific GUID for security.
3. Ascent360 stores the raw data securely.
4. The data is parsed into structured fields (using a predefined format).
5. Parsed data is loaded into the Ascent360 database.
6. Optional: Automated actions (e.g., sending an email) can be triggered based on the data.

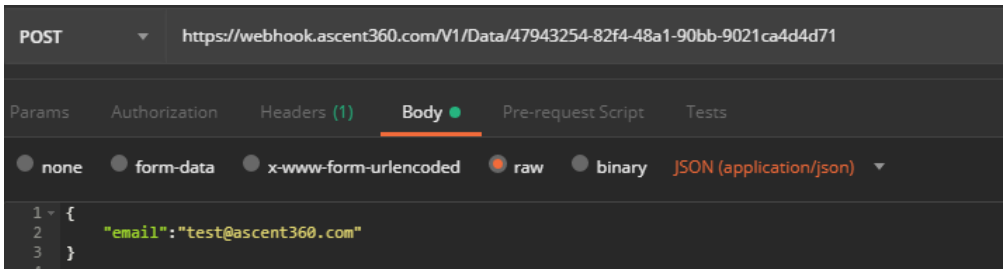
Setup Information

- Ascent360 configures your data source, assigning a **Client ID** and a **GUID** (used for security).
- Your unique webhook URL will be: `https://webhook.ascent360.com/V1/Data/{GUID}`

  Example: `https://webhook.ascent360.com/V1/Data/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`

- **Method**: Data must be sent via **HTTPS POST** to your assigned webhook URL.
- **Preferred Format**: JSON (XML is supported, but not preferred).
- **Implementation**: Can be done using **server-side** or **client-side** scripts.
- **Response**: The endpoint returns standard HTTP status codes only (200, 4xx, or 5xx).

How this may look in Postman:



How this may look in Code:

```
var client = new RestClient("https://webhook.ascent360.com/V1/Data/{GUID}");
var request = new RestRequest(Method.POST);
request.AddHeader("cache-control", "no-cache");
request.AddHeader("Content-Type", "application/json");
request.AddParameter("undefined", "{\n\t\"email\":\"test@ascent360.com\"\n}", ParameterType.RequestBody);
IRestResponse response = client.Execute(request);
```

Ascent360 processes data in real time using one of the following methods:

- **Generic JSON Parsing**: For simple, well-structured JSON.
- **Custom JSON Parsing**: For more complex JSON structures.
- **Custom Format Parsing**: For XML or other formats, based on predefined requirements.

Request Body Parameters

General Notes:

- Only valid JSON will be accepted.
- Even incorrect field names or aliasing won't throw errors if the JSON is valid.
- Values are **not case-sensitive**.
- Use the exact key names as shown in the **Example JSON Request** below.

Field	Include in Request Body As	Accepted Data Formats	Required?	Description
Source Name	SourceName	String	**Required**	Populates Source Name field in Ascent360.
Source Type	SourceType	String	**Required**	Populates Source Type field in Ascent360.
First Name	FirstName	String	**Recommended**	
Last Name	LastName	String	Optional	
Date of Birth	DateOfBirth	Date	Optional	mm-dd-yyyy
Gender	Gender	Text - "male" / "female"	Optional	Only “Male” and “Female” are stored. Other values display as “Unknown” in Ascent360.
Email	Email	Email (string)	**Recommended**	Email address.
Explicit Email Permission	EmailPermission	Boolean - true / false	**Recommended**	Denotes a change in explicit opt in / opt out from email communication. **Exclude if explicit consent is not given or withdrawn.** TRUE = email marketing explicit opt-in. FALSE = email marketing explicit opt-out.
Phone	Phone	International phone (string)	Optional	Phone number. Explicit Phone Permission (separate field) is applied to this field.
Mobile	Mobile	international phone (string)	Optional	Mobile phone number. Explicit SMS Permission (separate field) is applied to this field.

Explicit Phone Permission	PhonePermission	Boolean - true / false	Optional	Denotes a change in explicit opt in / opt out for phone communication. Permissions are applied to the "Phone" field, not “Mobile”. **This field is not used for SMS / Mobile communications. Exclude if explicit consent isn’t given or withdrawn.**
Explicit SMS Permission	SMSPermission	Boolean - true / false	Optional	Denotes a change in explicit opt in / opt out for SMS. communication. **Exclude from your payload if explicit consent is not given or withdrawn.** TRUE = SMS explicit opt-in. FALSE = SMS explicit opt-out.
Address	Address	JSON object		The address block must be passed within the address JSON object. See previous example.
ADDRESS LINE 1	Addressline1	String	Optional	Field in address block
ADDRESS LINE 2	Addressline2	String	Optional	Field in address block
CITY	City	String	Optional	Field in address block
STATE	State	String	Optional	Field in address block
ZIP	Zip	String	Optional	Field in address block
COUNTRY	Country	String	Optional	Field in address block
Explicit Address Permission	AddressPermission	Boolean - true / false	Optional	Denotes a change in explicit opt in / opt out from contact by mail. **Exclude if explicit consent isn’t given or withdrawn.**
Activities	Activities	Array - text values	Optional	An array of text values. Please contact your CSM if you anticipate needing this field.
Interests	Interests	Array - text values	Optional	An array of text values. Please contact your CSM if you anticipate needing this field.

Newsletters	NewsLetters	Array - text values	Optional	An array of text values. Please contact your CSM if you anticipate needing this field.
Provider Type	ProviderType	numeric	Optional	This is the ID of the provider you want to send contacts to – i.e. Acoustic is "7". Mailchimp is "3". If the data *just* needs to flow into Ascent360 (and no other location) you do not need to include this field.
Provider List IDs	ListIds	Array - text / numeric values	Optional	This is an array of lists a form submitter should be added to *__within__* the provider (defined by provider ID). (i.e. ID of newsletter list).

Example JSON HTTP Request Body

```
{
  "SourceName": "Email Signup Form",
  "SourceType": "Webforms",
  "FirstName": "James",
  "LastName": "Smith",
  "DateOfBirth": "12-01-1980",
  "Gender": "Male",
  "Email": "ABC@test.com",
  "EmailPermission": "true",
  "Phone": "19801234567",
  "Mobile": "19801234567",
  "PhonePermission": "false",
  "SMSPermission": "true",
  "Address": {
    "addressline1": "444 South",
    "Addressline2": "123 Maple Street",
    "city": "Pretendville",
    "state": "NY",
    "zip": "12345",
    "country": "USA"
  },
  "AddressPermission": "false"
}
```

If implementing in the web browser, create a JSON string using JavaScript then call the webhook via an XMLHttpRequest. Only the POST method will work. Alternately, this can be accomplished through server-side scripting (i.e. C#, Java, Rust, etc.).

```
var xhttp = new XMLHttpRequest();
xhttp.open("POST", "  --insert custom webhook URL here--  ");
xhttp.send(     --insert JSON string here--      );
```

Do’s & Don’ts

- ✅ Use correct field names and structure.
- ❌ Don’t assume a 200 response means success—data may not be parsed/stored if the format is incorrect.
- ✅ JSON is case-insensitive, but it's best to match key names to the documented format.

Retry Requirement

To prevent data loss during temporary service interruptions—which are normal for any cloud platform—clients must configure automatic retries for webhook calls on their side (wherever the webhook is triggered). If retries are not in place, any brief downtime can result in missed data. A common best practice is to attempt **3–5 retries within about 15 minutes**, spacing them out by a few minutes each time. This ensures that most transient issues are resolved without requiring manual intervention. After the final retry, log the failure so it can be reviewed later.

Why This Matters

Webhooks are designed for real-time communication, but even the most reliable systems experience occasional downtime or network hiccups. Without retries, any single failure means lost data—and that can impact reporting, customer experience, and downstream processes. Automatic retries dramatically reduce this risk and keep your integration resilient.

"Missed Data"

If any records did not land successfully, you can resend any previously failed records **at any time** (to the webhook URL) — we can process them as soon as they’re pushed to us.