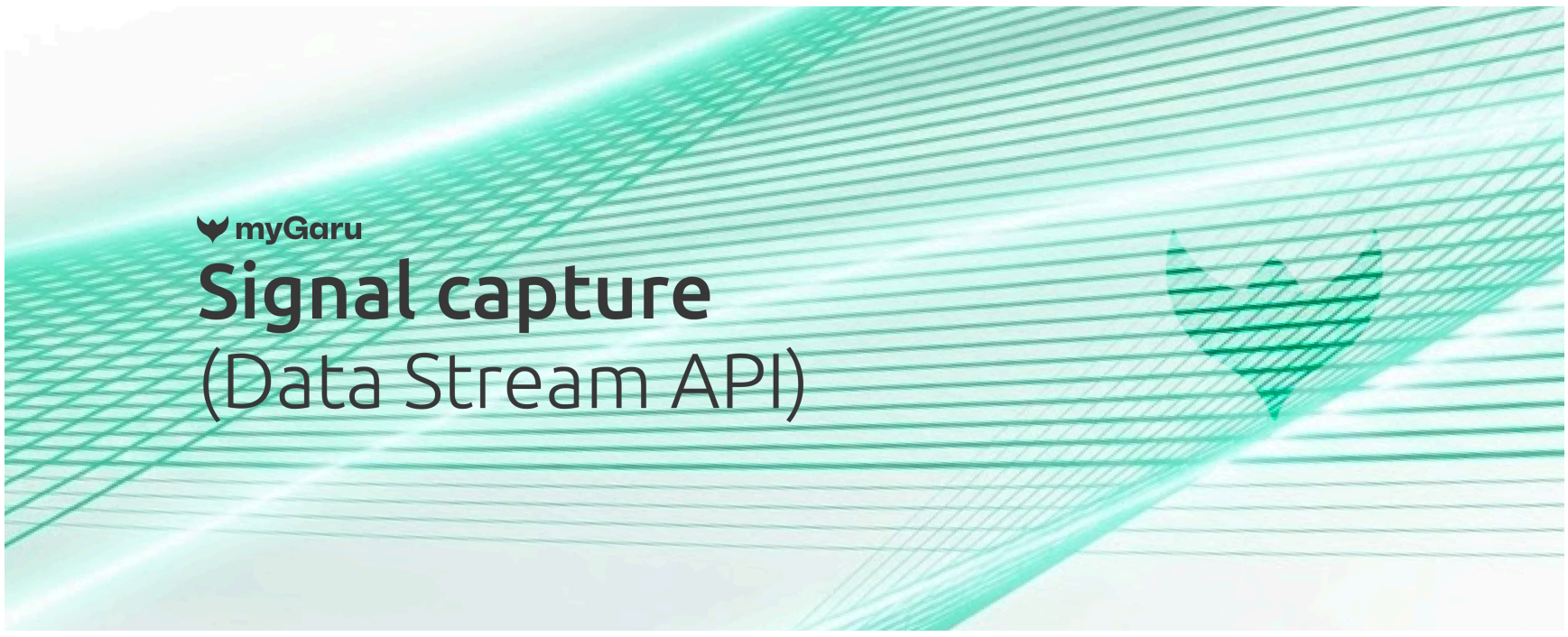


This document explains how the myGaru Data Stream API captures real-time signals from websites and mobile applications, appends them to existing customer profiles, and makes these signals available for audience segmentation and analytics within the DCR Client. It also outlines the setup flow, administrative configuration, and the API calls required by implementers.

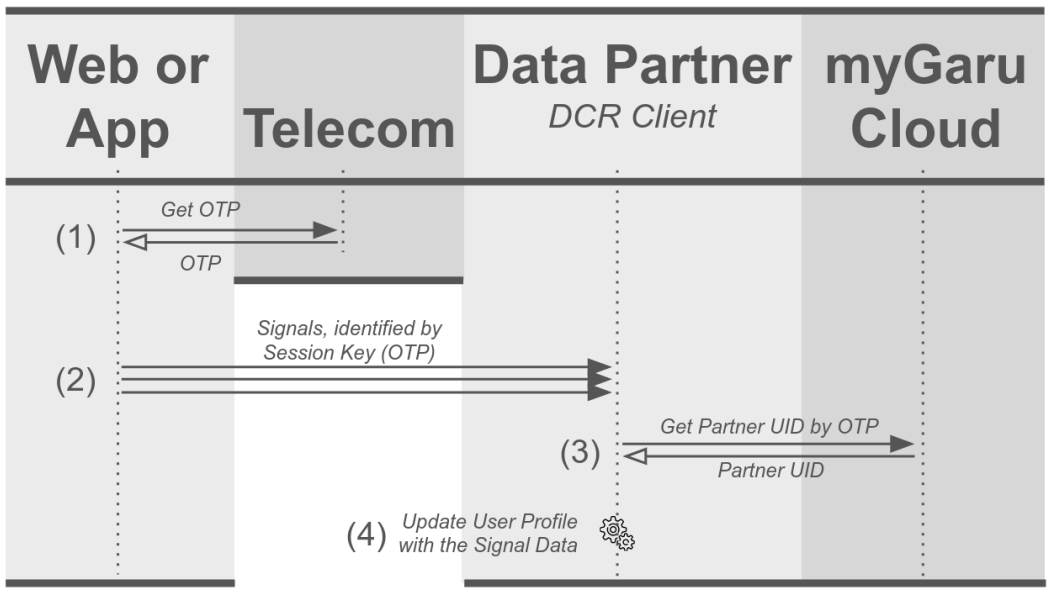
Sections at a glance

- **Chapter 1 — General Description** — purpose of the API, how it sits inside the client-side DCR component, and how captured signals are later used for segmentation and analytics.
- **Chapter 2 — Data Stream API Setup Process** — high-level steps to configure the endpoint, generate code, define sources and event fields, implement tracking, obtain an OTP, submit events, and create audience segments.
- **Chapter 3 — Data Model in the Data Stream API** — definition of supported data types (Label, Numeric, Boolean) and basic formatting rules with simple examples.
- **Chapter 4 — Data Stream API Configuration** — administrative settings in the web UI: API endpoint (FQDN), code generation, allowed domains.
- **Chapter 5 — Data Structure Configuration** — where and how profile fields are defined (name, type) and handed over to developers together with the profile identifier.
- **Chapter 6 — API Calls** — session retrieval (get_OTP) and operations for Label, Numeric and Boolean data; batch updates via a single request.
- **Chapter 7 — JavaScript SDK** — gives the information about available JS SDK and how to use it.
- **Chapter 8 — Cases** — worked examples, including abandoned-cart retargeting and interest-based targeting in a retailer’s mobile application.

1. General Description

The myGaru **Data Stream API** is a mechanism provided to myGaru partners for collecting real-time data signals about user behaviour on their website or mobile application and adding the gathered information to the profiles of existing customers. The collected data can later be used to build audience segments for both targeting and analytics purposes. All data is collected using a **library** that is installed on the partner’s website or in their mobile application.

The Data Stream API is an integral part of the client-side component of myGaru platform called **DCR Client.** It functions as part of the DCR Client’s capabilities. The overall interaction model is shown in the following diagram:



2. Data Stream API Setup Process

The DS API setup process gives developers and integrators a high-level understanding of how the Data Stream API is configured and how its components interact across systems.

Integration Step	myGaru DCR side	Website / Mobile App side
1. API Endpoint Setup	Specify the IP addresses of the Data Stream API Endpoint	
2. Code Generation	Generate JavaScript code or provide instructions for SDK/API integration	Integrate the code into the website or application
3. Specify domains as Referrer	Specify the list of domains, which can initiate signals	
4. Column Properties Configuration	On the Builder screen create columns: Name, Type (Text/Numeric/Boolean), Tag (optional)	Obtain field names
5. Event Tracking Logic		Implement logic for tracking specific events (e.g. abandoned cart, product interest)
6. Events Submission		Use API calls to update data
7. Segment Creation	Create audience segment in DCR based on the received tracking events	

Once the overall setup steps of the Data Stream API are outlined, a more detailed examination of its functionality follows, including data types, configuration settings and API operations.

3. Data Model in the Data Stream API

Data Stream API Data Types:

- **Text** — an event attribute in a text format. For example, it can be “Motorola” for the column “Phone” or “Greece,Thailand” for the column “Destinations of interest”
- **Numeric** — an event attribute in a numeric format representing a measurable quantity, such as the order amount or the number of visits to certain sections of the site, for example 10 or 89.99
- **Boolean** — an event attribute in a boolean format that identifies a parameter as either true or false.

4. Data Stream API Configuration

Management of Data Stream API parameters is done through a web interface, which includes the following settings:

General Settings

- **Data Stream API Endpoint IP Addresses:** as the name suggests, this is a list of IP addresses, where DCR Client serves incoming signals.
- The **“JavaScript Code”** area (filled only if the IP Addresses field is filled) contains the activation code for the Data Stream API client library, which can be placed on a website or integrated into an application.

5. Data Structure Configuration

Management of data structures of Data Stream API is performed in the **My Audience** section of myGaru Platform UI. For each profile, fields with corresponding parameters must be specified to store events. Fields are described by two parameters:

- **Name:** Consists of alphanumeric characters of the English alphabet and the underscore character ('_')
- **Type:** Can have one of the following values: Text, Numeric, Boolean
- **Tag:** optional parameter, used for filtering purposes in the WebUI only

Once the data structures are ready (created, updated, etc.), their information along with the profile identifier is provided to the developers of the website or mobile application. These developers will fill the fields with values from different sections of the website or application, using the appropriate API calls.

6. API Calls

Session Parameters

- session identifier (OTP) is the value, stored and available as the browser cookie `iuid`
- session API endpoint is an address of DCR Client, which listens to Data Stream API Calls. It can be obtained from the myGaru WebUI and looks like `PartnerID.signals.mygaru.com` where PartnerID is Data Vendor's identifier inside the myGaru platform, e.g. `https://20f079e4-7b79-4783-bf7a-0b936367c261.signals.mygaru.com`

Return status

All API calls return the following statuses:

- 2xx if request validated and accepted
- 400 if request cannot be validated
- 401/403 if request is not allowed

Operations on Text Data

Set text value

`GET {baseUrl}/data-stream/set-label?otp={iuid}&name={name}&label={value}`

Example:

- `GET {baseUrl}/data-stream/set-label?otp={iuid}&name="destinations of interest"&label="France,Greece"` will set the "destinations of interest" to "France,Greece" value.
- `GET {baseUrl}/data-stream/set-label?otp={iuid}&name="destinations of interest"&label=""` will clear the value of "destinations of interest".

Add text to the value

`GET {baseUrl}/data-stream/add-label?otp={iuid}&name={name}&label={value}&delimiter={value}`

Delimiter can be either empty or missing (which implies empty) defines how added value will be separated in a text column from previous value.

Note: if the field is empty, delimiter is ignored.

Example:

- `GET {baseUrl}/data-stream/add-label?otp={iuid}&name="destinations of interest"&label="Thailand"&delimiter=","` will add ",Thailand" to the value of "destinations of interest". With the example above, the final value will be the "France,Greece,Thailand"

Operations on Numeric Data

Numeric values are floating point, so 10.25 + 0.01 = 10.26

Set numeric value

`GET {baseUrl}/data-stream/set-number?otp={iuid}&name={name}&value={value}`

Example:

- `GET {baseUrl}/data-stream/set-number?otp={iuid}&name="cart total"&value=10.25`
  will set the “cart total” field to the 10.25

Update numeric value

`GET {baseUrl}/data-stream/step-number?otp={iuid}&name={name}&value={step}`

Changes the specified value by the `step` value. Positive step - increase the value, negative step - reduce the value.

Note: if field is empty, the value will be set to the `step` value.

Operations on Boolean Data

Set boolean value

`GET {baseUrl}/data-stream/set-boolean?otp={iuid}&name={name}&value={value}`

where value can be “true” or “false”.

7. JavaScript Software Development Kit (SDK)

Requirements: Node.js ≥ 20.

Initialisation

Use the initialisation script in the <head> section of the site:

```
<script>
  (function (w, d, n, u, c) {
    var t = w[n] || {};
    t.q = t.q || [];
    w[n] = new Proxy(t, {
      get: function (o, k) {
        if (k === "q") return o.q;
        if (k === "then" || typeof k === "symbol" || k === "Error") return;
        if (Object.prototype.hasOwnProperty.call(o, k)) return o[k];
        return function () {
          var a = [k];
          for (var i = 0; i < arguments.length; i++) a.push(arguments[i]);
          o.q.push(a);
        };
      },
    });
    var e = d.createElement("script");
    e.async = true;
    e.src = u;
    e.onload = function () {
      w[n].init(c);
    };
    var f = d.getElementsByTagName("script")[0];
    f.parentNode.insertBefore(e, f);
  })(
    window,
    document,
    "DataStreamApiClient",
    "https://dsalib.mgaru.dev/browser.global.js",
    { baseUrl: "https://[client_id].signals.mygaru.com" }
  );
</script>
```

where `baseUrl` need to be updated with Web-site owner’s (Data Vendor) PartnerID.

Note: it is better to copy this script from the myGaru WebUI, where it is always updated to the actual configuration.

SDK calls

All SDK calls match REST API calls, described above:

- **await** dsClient.setText(parameter, value)
  - e.g. (“destinations of interest”, “France,Greece”)
- **await** dsClient.addText(parameter, value, delimiter)
  - e.g. (“destinations of interest”, “Thailand”, “,”)
- **await** dsClient.setNum(parameter, value)
  - e.g. (”cart sum”, 25.10)
- **await** dsClient.stepNum(parameter, value)
  - e.g. (”cart sum”, 0.9)
- **await** dsClient.setBool(parameter, boolean)
  - e.g. ("abandoned_cart", true)

8. Cases

Use case description:	An e-commerce platform wants to run a retargeting ad campaign for its audience who have abandoned their shopping carts.
Actors:	**E-commerce platform** (advertiser)
Preconditions:	• E-commerce platform is a registered partner of the myGaru platform • In the myGaru web interface, the e-commerce platform configures: ◦ Data Stream API Endpoint (FQDN) ◦ JS code is generated or an integration specification is provided • In the My Audience section of myGaru Platform, the e-commerce platform creates a field: ◦ abandoned_cart, type: Boolean
Main flow:	• The e-commerce platform implements the logic to detect the event "user abandoned the cart" (abandoned_cart) • After that, the e-commerce platform calls: ◦ `set-boolean (OTP, "abandoned_cart", true)` • The event is registered via myGaru Data Stream API in real time • A segment is created in DCR based on the condition: ◦ abandoned_cart = "true" ◦ Access Level: Private • In any DSP integrated with myGaru, the e-commerce platform creates an ad campaign targeting this segment: “Users who abandoned cart”
Postcondition:	The e-commerce platform has launched a retargeting ad campaign for its own audience of users who abandoned their shopping carts without completing a purchase.

Use case description:	A retailer wants to run an ad campaign targeting mobile application users who showed interest in electronics and gadgets.
Actors:	**Retailer**
Preconditions:	• The Retailer is a registered partner of the myGaru platform • In the myGaru web interface, the following is configured: ◦ Data Stream API Endpoint (FQDN) ◦ JS code is generated and integrated within the Retailer’s mobile application • In the My Audience section of the myGaru Platform, the following field is created: ◦ interests, type: Label
Main flow:	• The mobile application implements the logic to track user interest in specific categories (namely: viewing products in the “Electronics” or “Gadgets” section) • After that, the Retailer application calls: ◦ `add-label(OTP, "interests", "electronics")` ◦ `add-label(OTP, "interests", "gadgets")` • The event is registered via myGaru Data Stream API in real time • In the My Audience section, the Retailer creates an audience segment based on the condition: ◦ interests contain "electronics" or "gadgets"

	◦ Access Level: Private (only for the Retailer's own usage) • In any DSP connected with myGaru, the Retailer creates an ad campaign targeting this segment: “Interest in electronics and gadgets”
Postcondition:	The Retailer has launched a personalised ad campaign targeting mobile application users who have shown interest in electronics or gadgets.