Technical Overview

The Realtime CRM Event functionality is perfect if you want to react quickly to events originating from your system, and need data in Symplify to be as up-to-date as possible at every single point in time. It's ideal if you already have a message broker where you publish application events, or if you want to avoid explicit integration with the Symplify REST API.

Note that the Realtime CRM Event functionality is **not** intended or suitable for bulk operations, e.g. daily user data synchronization or bulk message generation. For these types of requirements, we recommend using the Batch Import and Batch Transactional functions.

In a Realtime CRM event setup, Symplify is configured to listen for messages from a Message Broker (SQS or RabbitMQ), and instructed to take actions depending on incoming messages. As long as the incoming message format follows the recommended format described below, no additional integration is required. If your message broker produces messages in another format, this will require custom implementation on our side, along with additional integration costs.

Configuration

The configuration is made using the Symplify admin interface and includes:

1. Endpoint identifier/URI/Queue identifier
2. Authentication information
    a. For RabbitMQ this will be username/password for a user with read access to the specified queue
    b. For SQS, this will be the key + secret key of an AWS IAM API user with read access to the specified queue. Alternatively, you can grant read access on your queue to our AWS account, which completely eliminates the need for credentials registration in Symplify.
3. Message type -> action mapping (see below)

Event Processing

Events are processed in the order they are received from the message broker. We will rely on the message broker to ensure proper ordering and "only once" delivery of events. This means that if the broker delivers the exact same event message more than once ("at least once" message delivery), both of these events will be processed. This is something that needs to be taken into account when setting up this kind of integration. We recommend using a message broker that supports ordered, "only once" message delivery.

Events and Format

Incoming event payloads need to be in JSON format. We highly recommend using the default event format described below.

If you find it impossible to produce events in the default event format, Symplify can implement a custom message converter at additional costs.

All messages that you want a custom Symplify message converter to consume will need to be typed either through a header attribute or a message attribute, depending on the technology used by your message queue. Your custom event format will be transformed using the custom-built converter to the standard events described below. Consequently, for us to be able to implement a custom converter, we do need the events from your system to contain all the necessary information needed to populate our standard event objects.

The following events are currently available (more events may be added in the future). You may choose to use only a subset of these in your particular integration:

1. USER_REGISTRATION,
2. USER_UPDATE,
3. USER_BLOCKED,
4. USER_CONSENT_UPDATE,
5. WITHDRAWAL_REQUESTED,
6. WITHDRAWAL_APPROVED,
7. WITHDRAWAL_REJECTED,
8. WITHDRAWAL_CANCELLED,
9. DEPOSIT_REQUESTED,
10. DEPOSIT_APPROVED,
11. DEPOSIT_REJECTED,
12. DEPOSIT_CANCELLED

The events are a high-level representation that something has happened in your system. On the Symplify side, a specific event type will be mapped to one or more Actions. An example would be a “USER_REGISTRATION” action in your system, which we may use to a "Create contact" action followed by a "Enter journey" action to add the user to the onboarding Journey. This is all determined using configuration, and can be changed dynamically in runtime.

Symplify Actions

To each of the above events relevant in your integration, you can attach a series of actions for Symplify to perform. The actions currently available and their configuration are as follow:

1. Create contact
    a. List id
    b. Default email subscription status (Defaults to true)
    c. Default mobile subscription status (Default to true)
2. Update contact
    a. List id
3. Set property
    a. List id
    b. Property id
    c. Property Value
4. Create purchase history row
    a. List Id
5. Enter journey
    a. Journey Id
6. Anonymize contact
    a. List Id
    b. Find by (Original Id or Email Address)
7. Temp. block contact
    a. List Id
    b. Find by (Original Id or Email Address)
    c. Block for (unit)
    d. Block for time type(minutes, hours, days, months, years)

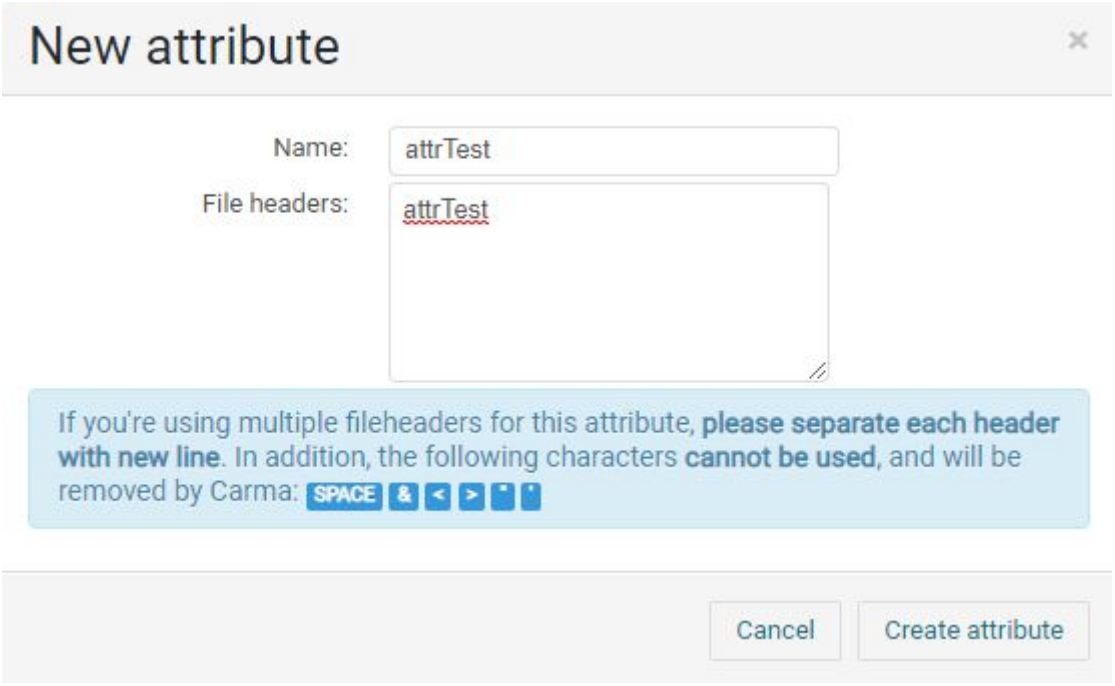
Mapping your Message Data to Symplify Attributes

In order to map the data we receive from your messages to Symplify attribute, you will need to create a recipient attribute for each field you wish to populate. For each of these fields, you will set the file header to the same name as the attribute's name in your message.

Example: Given the following payload:

```
{
...
  "properties": {
    "attrTest": "Some value"
  }
}
```

You will create an attribute like so:



Default Message Format

The simplest possible integration with Symplify can be achieved using the default message format described below. There is no requirement that you need all the different message types; a subset can be used if this is all you need for the particular integration.

USER_REGISTRATION

```
{
  "type": "USER_REGISTRATION",
  "contactId": "abc123",
  "properties": {
    "prop1": "value1",
    "prop2": "value2"
  }
}
```

- contactId: Unique shared identifier of the user between your system and Symplify
- properties: Information that should be registered for the user in Symplify. Properties can include any attribute mapped as described above.

USER_UPDATE

```
{
  "type": "USER_UPDATE",
  "contactId": "abc123",
```

```
  "properties": {
    "prop1": "value1",
    "prop2": "value2"
  }
}
```

- contactId: Unique shared identifier of the user between your system and Symplify
- properties: Information that should be registered for the user in Symplify. Properties can include any attribute mapped as described above.

USER_BLOCKED

```
{
  "type": "USER_BLOCKED",
  "contactId": "abc123",
  "properties": {
    "prop1": "value1",
    "prop2": "value2"
  }
}
```

- contactId: Unique shared identifier of the user between your system and Symplify
- properties: Information that should be registered for the user in Symplify. Properties can include any attribute mapped as described above.

USER_CONSENT_UPDATE

```
{
  "type": "USER_CONSENT_UPDATE",
  "contactId": "abc123",
  "channel": "EMAIL",
  "consented": true,
  "properties": {
    "prop1": "value1",
    "prop2": "value2"
  }
}
```

- contactId: Unique shared identifier of the user between your system and Symplify
- properties: Information that should be registered for the user in Symplify. Properties can include any attribute mapped as described above.
- channel: The channel that this consent applies to (one of EMAIL, SMS, PUSH, INAPP, PRINT, PAGE, or AUDIO).
- consented: Whether the user opted in or out.

WITHDRAWAL_REQUESTED

```
{
  "type": "WITHDRAWAL_REQUESTED",
  "contactId": "abc123",
  "timestamp": "2020-02-05T13:20:00",
```

```
  "amount": 100.50,
  "transactionId": "abc123",
  "vendor": "someVendor",
  "paymentType": "VISA",
  "properties": {
    "prop1": "value1",
    "prop2": "value2"
  }
}
```

- contactId: Unique shared identifier of the user between your system and Symplify
- properties: Information that should be registered for the user in Symplify. Properties can include any attribute mapped as described above.
- timestamp: ISO-8601 formatted date string.
- amount: The amount
- transactionId: Unique transaction id from your system (optional)
- vendor: Vendor identifier from your system (optional)
- paymentType: payment type identifier (optional)

WITHDRAWAL_APPROVED

```
{
  "type": "WITHDRAWAL_APPROVED",
  "contactId": "abc123",
  "timestamp": "2020-02-05T13:20:00",
  "amount": 100.50,
  "transactionId": "abc123",
  "vendor": "someVendor",
  "paymentType": "VISA",
  "properties": {
    "prop1": "value1",
    "prop2": "value2"
  }
}
```

- contactId: Unique shared identifier of the user between your system and Symplify
- properties: Information that should be registered for the user in Symplify. Properties can include any attribute mapped as described above.
- timestamp: ISO-8601 formatted date string.
- amount: The amount
- transactionId: Unique transaction id from your system (optional)
- vendor: Vendor identifier from your system (optional)
- paymentType: payment type identifier (optional)

WITHDRAWAL_REJECTED

```
{
  "type": "WITHDRAWAL_REJECTED",
```

```
  "contactId": "abc123",
  "timestamp": "2020-02-05T13:20:00",
  "amount": 100.50,
  "transactionId": "abc123",
  "vendor": "someVendor",
  "paymentType": "VISA",
  "properties": {
    "prop1": "value1",
    "prop2": "value2"
  }
}
```

- contactId: Unique shared identifier of the user between your system and Symplify
- properties: Information that should be registered for the user in Symplify. Properties can include any attribute mapped as described above.
- timestamp: ISO-8601 formatted date string.
- amount: The amount
- transactionId: Unique transaction id from your system (optional)
- vendor: Vendor identifier from your system (optional)
- paymentType: payment type identifier (optional)

WITHDRAWAL_CANCELLED

```
{
  "type": "WITHDRAWAL_CANCELLED",
  "contactId": "abc123",
  "timestamp": "2020-02-05T13:20:00",
  "amount": 100.50,
  "transactionId": "abc123",
  "vendor": "someVendor",
  "paymentType": "VISA",
  "properties": {
    "prop1": "value1",
    "prop2": "value2"
  }
}
```

- contactId: Unique shared identifier of the user between your system and Symplify
- properties: Information that should be registered for the user in Symplify. Properties can include any attribute mapped as described above.
- timestamp: ISO-8601 formatted date string.
- amount: The amount
- transactionId: Unique transaction id from your system (optional)
- vendor: Vendor identifier from your system (optional)
- paymentType: payment type identifier (optional)

DEPOSIT_REQUESTED

```
{
  "type": "DEPOSIT_REQUESTED",
  "contactId": "abc123",
  "timestamp": "2020-02-05T13:20:00",
  "amount": 100.50,
  "transactionId": "abc123",
  "vendor": "someVendor",
  "paymentType": "VISA",
  "properties": {
    "prop1": "value1",
    "prop2": "value2"
  }
}
```

- contactId: Unique shared identifier of the user between your system and Symplify
- properties: Information that should be registered for the user in Symplify. Properties can include any attribute mapped as described above.
- timestamp: ISO-8601 formatted date string.
- amount: The amount
- transactionId: Unique transaction id from your system (optional)
- vendor: Vendor identifier from your system (optional)
- paymentType: payment type identifier (optional)

DEPOSIT_APPROVED

```
{
  "type": "DEPOSIT_APPROVED",
  "contactId": "abc123",
  "timestamp": "2020-02-05T13:20:00",
  "amount": 100.50,
  "transactionId": "abc123",
  "vendor": "someVendor",
  "paymentType": "VISA",
  "properties": {
    "prop1": "value1",
    "prop2": "value2"
  }
}
```

- contactId: Unique shared identifier of the user between your system and Symplify
- properties: Information that should be registered for the user in Symplify. Properties can include any attribute mapped as described above.
- timestamp: ISO-8601 formatted date string.

- amount: The amount
- transactionId: Unique transaction id from your system (optional)
- vendor: Vendor identifier from your system (optional)
- paymentType: payment type identifier (optional)

DEPOSIT_REJECTED

```
{
  "type": "DEPOSIT_REJECTED",
  "contactId": "abc123",
  "timestamp": "2020-02-05T13:20:00",
  "amount": 100.50,
  "transactionId": "abc123",
  "vendor": "someVendor",
  "paymentType": "VISA",
  "properties": {
    "prop1": "value1",
    "prop2": "value2"
  }
}
```

- contactId: Unique shared identifier of the user between your system and Symplify
- properties: Information that should be registered for the user in Symplify. Properties can include any attribute mapped as described above.
- timestamp: ISO-8601 formatted date string.
- amount: The amount
- transactionId: Unique transaction id from your system (optional)
- vendor: Vendor identifier from your system (optional)
- paymentType: payment type identifier (optional)

DEPOSIT_CANCELLED

```
{
  "type": "DEPOSIT_CANCELLED",
  "contactId": "abc123",
  "timestamp": "2020-02-05T13:20:00",
  "amount": 100.50,
  "transactionId": "abc123",
  "vendor": "someVendor",
  "paymentType": "VISA",
  "properties": {
    "prop1": "value1",
    "prop2": "value2"
  }
}
```

- contactId: Unique shared identifier of the user between your system and Symplify
- properties: Information that should be registered for the user in Symplify. Properties can include any attribute mapped as described above.
- timestamp: ISO-8601 formatted date string.
- amount: The amount
- transactionId: Unique transaction id from your system (optional)
- vendor: Vendor identifier from your system (optional)
- paymentType: payment type identifier (optional)