BrandMaker

Message Based Integration Service

Manual

Version 7.3

EN

25. March 2022

Copyright

Rüppurrer Straße 1, 76137 Karlsruhe (Germany), www.brandmaker.com

Your feedback is important to us!

We would be grateful to be notified of any errors you may discover. Just send us an e-mail to documentation@brandmaker.com.

1 Introduction 5
1.1 Terminology 5
1.2 Additional Information 6
1.3 Structure of This Manual 6
2 Delimitation 7
2.1 Administration 7
2.2 Consumer Development 7
2.3 Context Delimitation 7
3 Architecture Overview 8
4 Administration: Consumer Registration for MBI 9
4.1 Create Consumer 11
4.2 Manage MBI Consumer 12
4.3 Test MBI Consumer 13
4.4 Trigger MBI Consumer Manually 14
5 Requirements for the REST API of the Consumer 15
5.1 Implementation of the REST Interface 15
5.2 Validation of the Request in the Consumer 15
5.3 Data Format of the Transaction Message 16
5.3.1 Description of Attributes 17
5.4 Notes on the Processing of Transaction Messages 17
6 Example Implementation of a Consumer 18
6.1 Basic Structure 18
6.2 General Notes 19
6.3 Signature Validation 19
6.4 Using the Sample Consumer on GitHub 20
7 Documentation of Available Transactions per Module 21
7.1 Job Manager 21
7.2 Marketing Planer 26
7.3 Review Manager 26
7.4 Resource Management 26
8 Security Advice 27
8.1 SSL Encryption 27
8.2 Request Validation 27
8.3 Requests to the BrandMaker API 27

1 Introduction

With Release 7.3, BrandMaker provides the “Message Based Integration Service” for the first time. This central service is used for the customer-side integration of the BrandMaker system into the customer’s infrastructure.

The service is based on the so-called “publish / subscribe” principle. Starting with the Job Manager module in Release 7.3, all modules in BrandMaker send information about user transactions in the system to the MBI. Subscribers to the Service may receive and process these Transaction Messages.

In this way, loose coupling of the BrandMaker system and its modules with third-party applications is possible. Furthermore, the extensive APIs of the BrandMaker system and its modules are available to third-party applications to read or modify further data.

1.1 Terminology

References within the table are shown as clickable links with a →.

Consumer	→Subscriber
Event	The →Transaction message sent to the Consumer contains a list “Events” – events that occurred in the system and the associated data (→Payload)
MBI	BrandMaker **M**essage **B**ased **I**ntegration Service, refers to the service as a whole.
Payload	User data of the message. The payload is module and event dependent
Producer	→Publisher
Publisher	Sender of a transaction, generally a module of the BrandMaker system such as the Job Manager or the Marketing Planner
Subscriber	A system registered at the MBI to receive a transaction message
Transaction	User triggered or also automatic event within the BrandMaker module, e.g. “a new task has been created in the Job Manager”.
Transaction message	Information about a transaction (that has taken place) that the MBI sends to registered subscribers or consumers
Webhook	Generally, a method for asynchronous data integration between individual, delimited software systems. Specifically, the service that provides such a function: a →Consumer registers with a Webhook to receive data from it.

1.2 Additional Information

Good introductions and background knowledge in the category “Publish / Subscriber” or “Event Driven Architecture” can be found in this collection of links:

- https://requestbin.com/blog/working-with-webhooks/
- https://en.wikipedia.org/wiki/Webhook
- https://aws.amazon.com/de/event-driven-architecture/
- https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch02.html
- https://www.amazon.de/Enterprise-Integration-Patterns-Designing-Deploying/dp/0321200683
- https://www.amazon.de/Building-Microservices-Designing-Fine-Grained-Systems/dp/1492034029
- https://www.amazon.de/Publish-Subscribe-Systems-Communications-Distributed-ebook/dp/B008D30LUK

1.3 Structure of This Manual

This manual describes the various aspects of MBI.

- Architecture overview in chapter 3
- Administration of consumer registrations in the BrandMaker system in chapter 4
- Requirements for the consumer API in chapter 2
- Example implementation of a consumer in chapter 6
- Documentation of the transactions and provided data of the modules starting from chapter 6.2
    a. Job Manager
    b. Marketing Planer
    c. Review Manager
    d. Resource Management

2 Delimitation

2.1 Administration

The reader must basically be familiar with the administration of the BrandMaker system. This manual describes the specific settings of the consumer registration. You need administrator access to your BrandMaker system to make the necessary settings.

Preconditions

1. Navigate to the *page > Administration > System Settings*.
2. Search for the element `brandmaker.mbi.url`
3. Review an existing URL. Otherwise, enter the missing URL in this field. You can get the correct URL from BrandMaker Support.

2.2 Consumer Development

We assume that the reader, respectively the developer on the customer side is familiar with technologies like HTTP, REST, and the programming of web services.

All given example listings are provided in Java, but there is no constraint and no preference for any programming language.

Furthermore, knowledge of the application and configuration of the BrandMaker system is required. The developer needs administrator access to the BrandMaker system to make the necessary settings.

In order to automatically retrieve and store data from the BrandMaker system using a third-party system, knowledge of the corresponding development environments and APIs of the third-party system is required. These are also not the subject of this documentation. The third-party system is always presented as a black box in this context.

2.3 Context Delimitation

The interfaces offered are used for near-real-time synchronization of data sets in the BrandMaker system with third-party applications. Other purposes are not supported.

3 Architecture Overview

The basic architecture of the BrandMaker MBI is as follows:

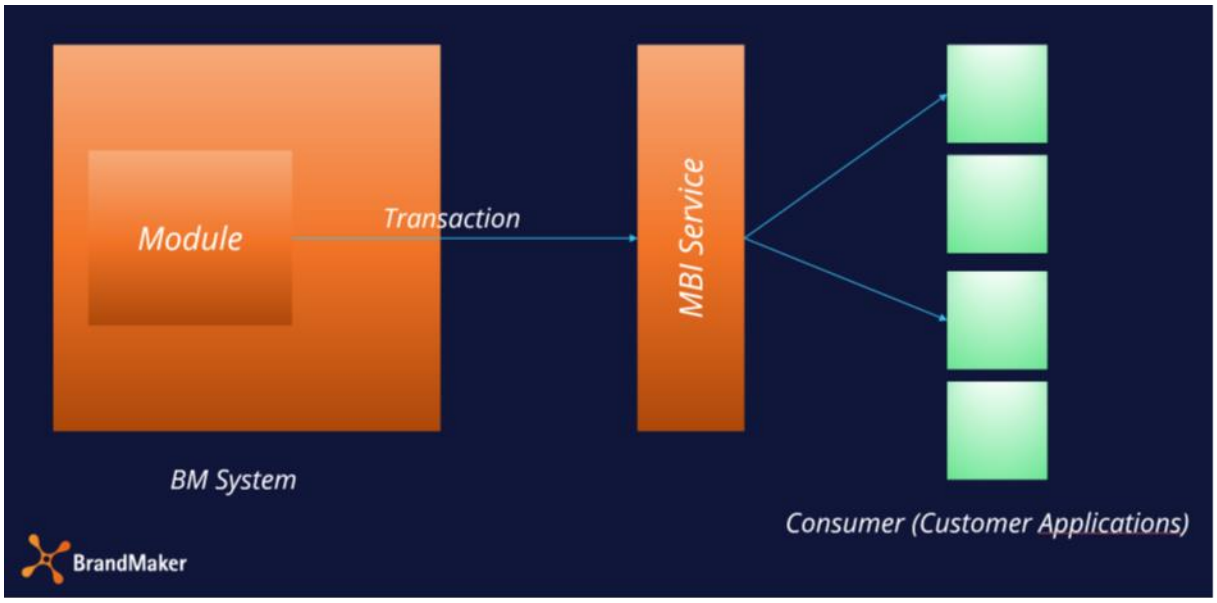


Figure 1 – Principle sketch BrandMaker MBI

The module sends information about completed transactions to the MBI. The MBI checks whether there is a registration for the module and the respective transaction and sends the information as well as a number of additional details about the source system and module to the registered third-party applications.

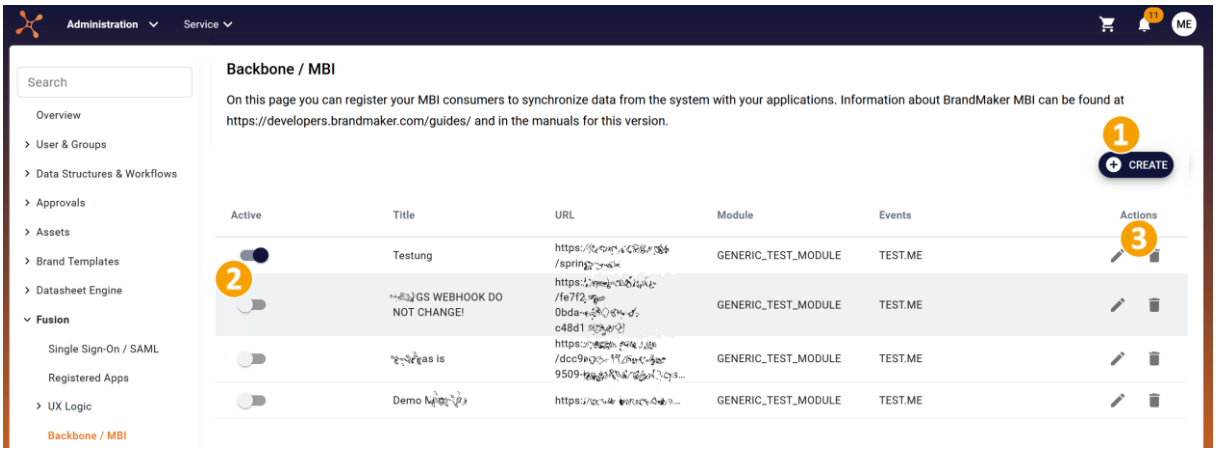
To do this, the third-party applications must provide a REST API capable of accepting the data sent and returning an acknowledgement of receipt.

The requirements for this REST API to be provided are described in chapter 5 later in this manual. Typically, the MBI sends the transaction data in JSON format to the registered consumer via HTTP POST request.

The registration of a consumer takes place in the respective customer system in the administration.

4 Administration: Consumer Registration for MBI

Log in to your BrandMaker system as an administrator to make the necessary settings. You can access the registration under *> Administration > Fusion > Backbone / MBI*:



Your screen view should now resemble the one above. Some screen texts may still be subject to change until the release version.

1 Create

Register the MBI Consumer, see 4.1.

3 Action buttons

In this area you can achieve the following actions:

- Edit MBI Consumer

  By clicking on the pencil icon, you can make changes to the configuration.
- Delete MBI Consumer

  After clicking on the trashcan icon, a confirmation prompt opens before you can fully delete the consumer.

2 Slider “Active”

Use this slider to enable or disable an MBI consumer.

Consumer characteristics

Input	Description
Event Timeout	How long do you wait for a response from the consumer? [unit s = seconds]; example: 30. After the timeout has elapsed, an error is registered. If too many errors occur in a row, the consumer will be disabled. See chapter 5.4
Retry delay	[unit s = seconds]; example: 60. MBI waits for the period of time specified here before attempting to redeliver the message.

Input	Description
Max Retries	Maximum number of attempts to send data after timeout.
Max Events	Maximum value of events that will be transmitted at once. These are sent in a batch and not individually.
Send missed messages	Missing transaction messages that could not be submitted will be submitted as soon as the Consumer is available again. During inactivity all transaction messages are cached.
Module and events	Here you select from which modules and events data should be received.

Note

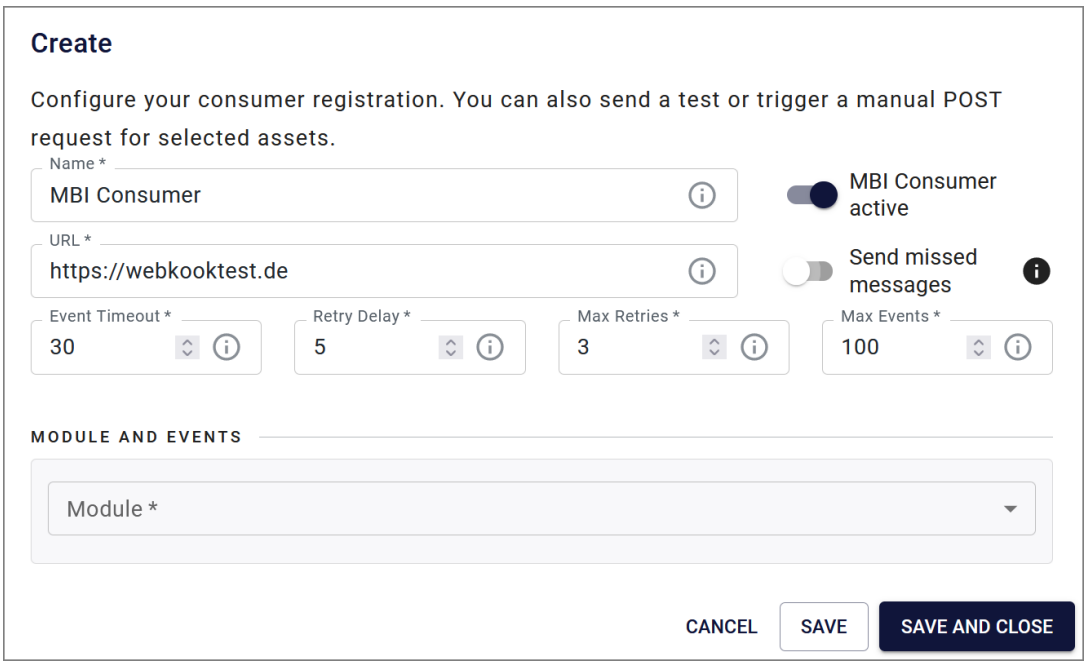
The maximum limit of the allowed input is for:

- *Event Timeout* = 900 seconds
- *Max. Retries* = 100
- *Max. Events* = 100

4.1 Create Consumer

Click *Create*.

The following dialog is displayed.

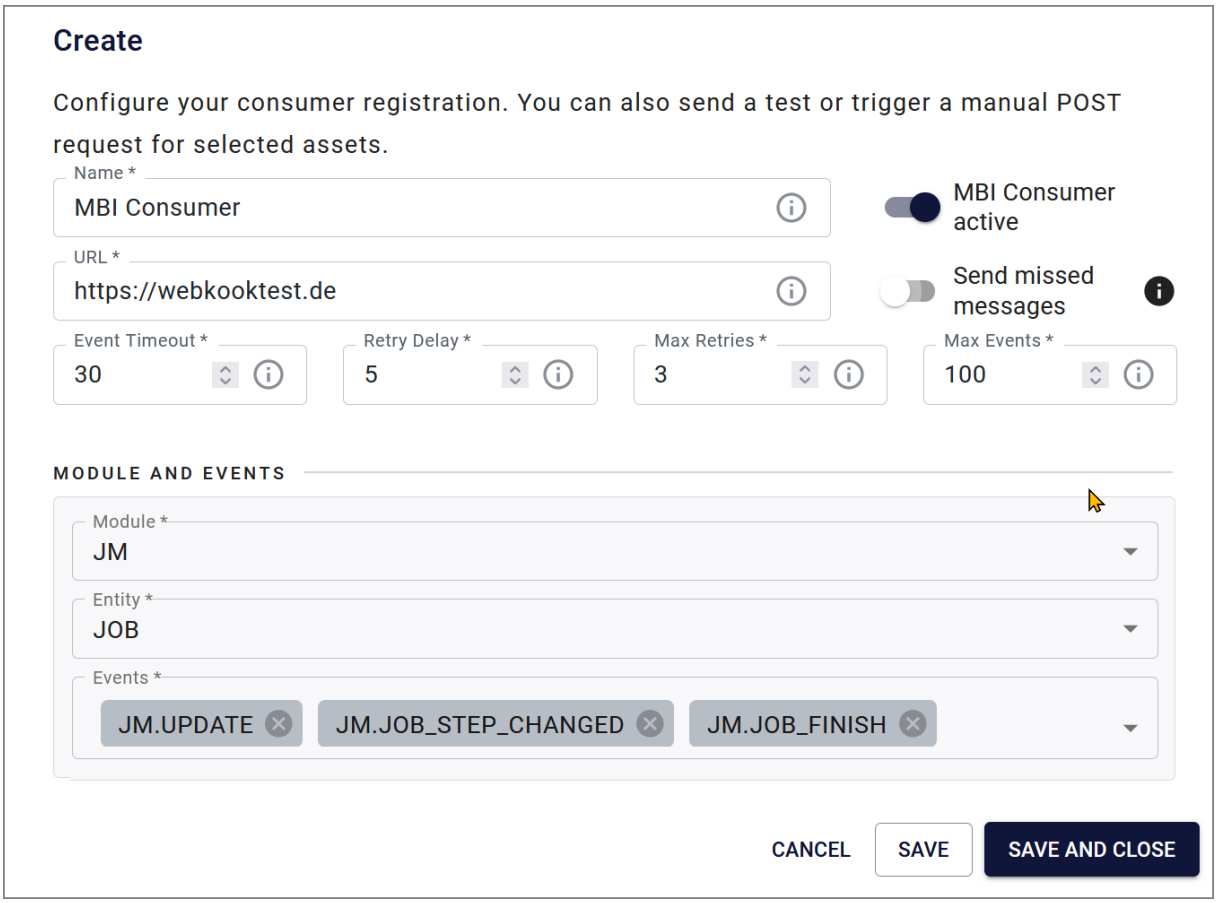


Edit the fields in the upper area. Required fields are marked with an asterisk (*). For a description of the fields, see the chapter 4.

Note: The consumer URL must be properly registered and configured for the HTTPS protocol. Experimental test environments with *localhost*-constructs do not work.

The *MBI Consumer active* slider is activated by default. If you do not want to do this because the environment is not finished configuring or you do not want to enable the consumer yet, save the configuration for the new consumer only after you have disabled the slider.

In the area *Module and Events* under the drop-down lists *Modules*, *Entity* and *Events*, specify which events should be reacted to:



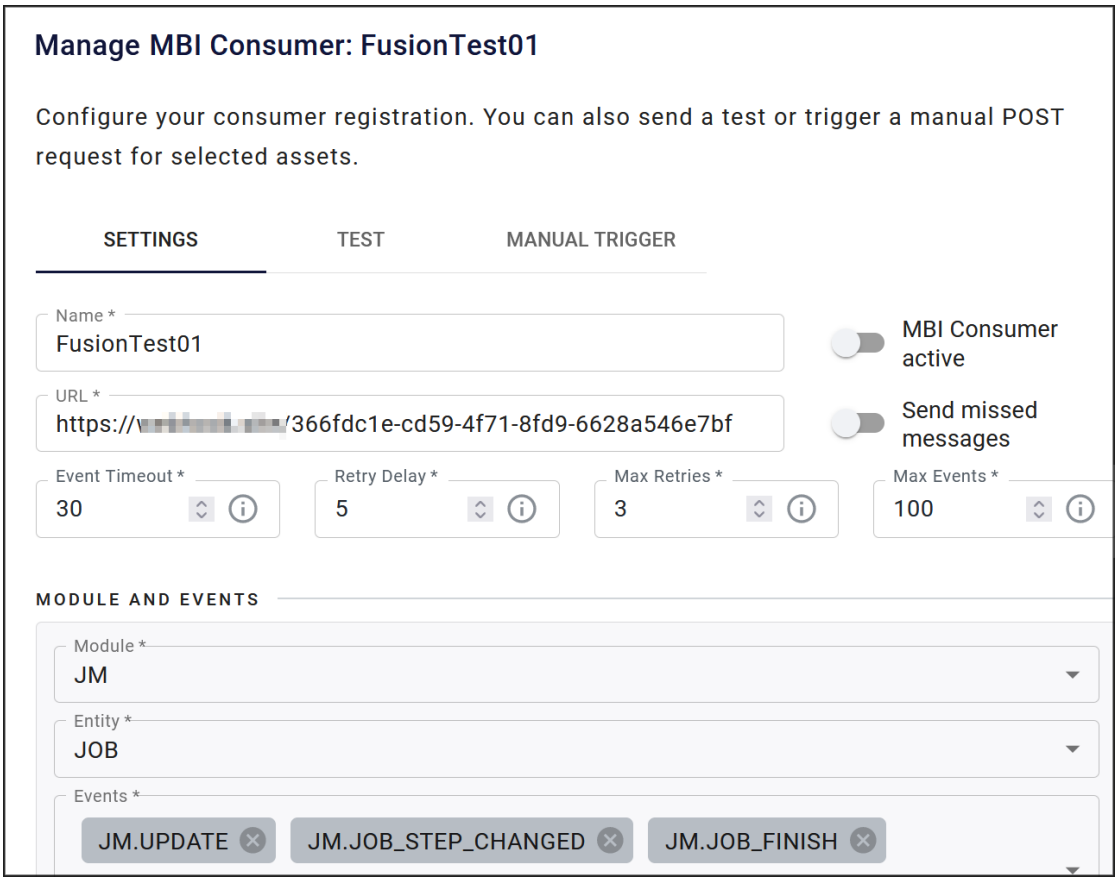
Click *Save* or *Save and Close*.

You have registered a Consumer.

4.2 Manage MBI Consumer

1. In the row of the Consumer you want to edit, click the pencil icon on the right side of the row.

The dialog *Manage MBI Consumer* appears:



2. Edit the properties of the consumer in the *Settings* tab.
3. Click *Save* or *Save and Close*.

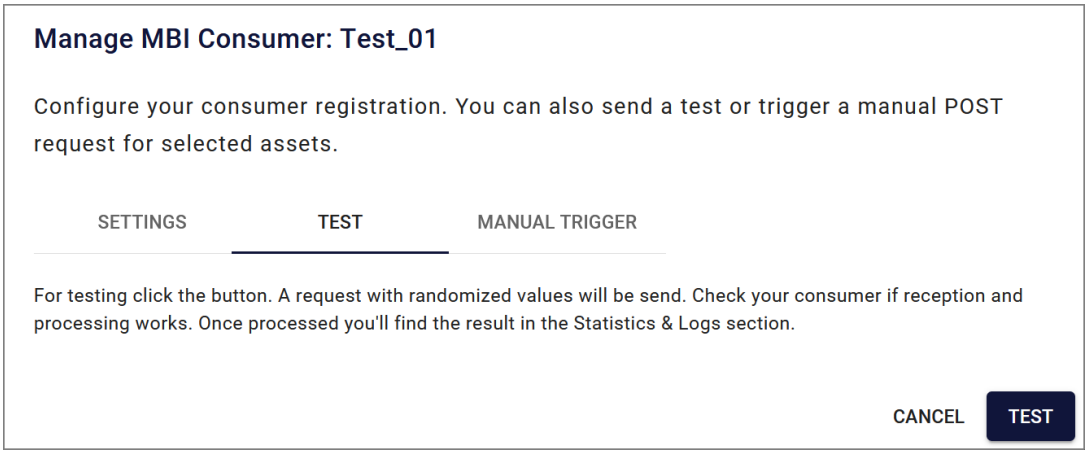
You have edited the MBI Consumer.

4.3 Test MBI Consumer

1. In the row of the Consumer you want to test, click the pencil icon on the right side of the row.

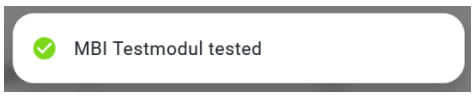
The dialog *Manage MBI Consumer* appears:

2. Go to the *Test* tab.



To test, click the *Test* button.

A request with random values is sent. Check if the reception and processing work with your consumer. If successful, a notification will be displayed as shown below.

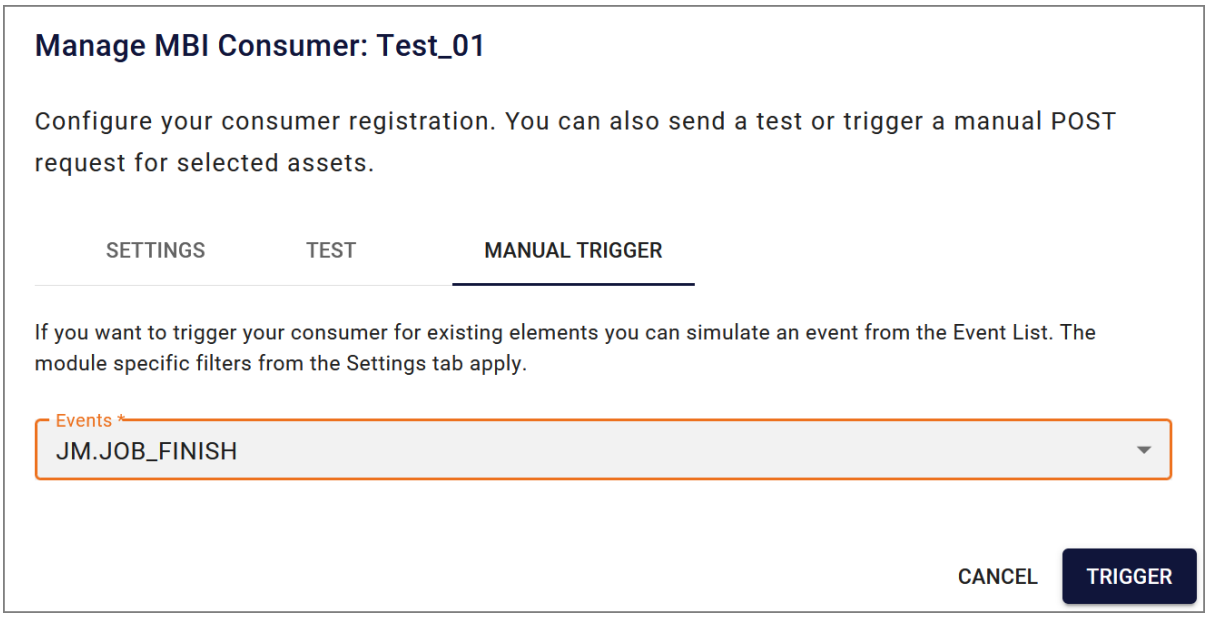


4.4 Trigger MBI Consumer Manually

If you want to trigger the consumer for existing elements, you can simulate an event from the event list.

1. In the row of the Consumer you want to test, click the pencil icon on the right.

The dialog *Manage MBI Consumer* appears:



2. Go to the *Manual Trigger* tab.
3. Select an event from the drop-down menu. The filters in the *Settings* tab apply.
4. Click the button *Trigger* to activate.
5. A fake message is generated to the consumer for testing purposes. Again, you will receive feedback in the form of an overlaid notification.

You can select the module *GENERIC_TEST_MODULE*, as the name suggests, to run generic tests.

5 Requirements for the REST API of the Consumer

An MBI Consumer must provide a REST interface to receive transaction messages, the URL of which is stored in the registry.

> **Note**
>
> Please note that this URL must be accessible via the public Internet. In addition, MBI only accepts SSL encrypted connections (https).

5.1 Implementation of the REST Interface

The MBI sends an HTTP POST request to the Consumer with the data described below. It must be able to receive and process this POST request.

A request can contain a number of transactions (see Attribute `events` in the data structure described below). The maximum number of messages transmitted in a batch is stored in the BrandMaker system administration when the Consumer is registered.

We recommend implementing only the POST method and rejecting all other HTTP methods with HTTP status 405 (“Method not allowed”).

For illustration, the example Consumer can be used, which is published on GitHub: https://github.com/brandmaker/MBI-Consumer.

5.2 Validation of the Request in the Consumer

Since this is asynchronous batch processing, it must be ensured that the request arriving at the Consumer was also sent by the BrandMaker instance and contains unaltered and correct data.

For this reason, a signature is sent in the request header, which can be used to validate the authenticity of the request in the Consumer. The request is a signed HTTP message according to https://tools.ietf.org/id/draft-cavage-http-signatures-07.html

The signature includes the entire request body, ensuring the integrity of the request as a whole. The validation of the message is then carried out in accordance with the standard described in the above RFC link and is not described further here.

To validate the signature, the public key of the BrandMaker system is required, which can be retrieved via the URL `https://<brandmaker system>/rest/so/keys/public` or `…/public-for-file`.

Recommendation for practice

We advise you to query this key once and store it statically in the application (Consumer). The key is generated once by BrandMaker and does not change for existing systems.

We strongly advise against picking up the key in the consumer dynamically via the `systemBaseUri`. This would allow an attacker to provide both public and private keys with a forged signature and submit spoofed requests this way!

5.3 Data Format of the Transaction Message

The transaction messages have a standardized data format. However, some of the attributes in the sent object depend on the context of the particular module. These are the yellow highlighted attributes in the JSON structure below.

The basic structure of the message is as follows:

```
{
  "systemBaseUri": "https://server",
  "customerId": "aaa-bbb-ccc",
  "systemId": "123-456-789",
  "events": [
    {
      "module": "MP",
      "operation": "update",
      "entity": "asset",
      "timestamp": 123456789,
      "objectId": {
        "assetId": 12345,
        "version": 4
      },
      "data": {
        "additionalProp1": {},
        "additionalProp2": {},
        "additionalProp3": {}
      },
      "userId": "string"
    },
    {
      "module": "MAPL",
      "operation": "update",
      "entity": "invoice-node",
      "timestamp": 123456789,
      "objectId": {
        "nodeId": 12345,
        "invoiceId": 4
      },
      "data": {
        "additionalProp1": {},
        "additionalProp2": {},
        "additionalProp3": {}
      },
      "userId": "string"
    }
  ]
}
```

5.3.1 Description of Attributes

Field	Type	Meaning
`systemBaseUri`	String	URL of the requesting BrandMaker system to make a callback against the APIs
`customerId`	String	Unique identifier for this customer
`systemId`	String	Unique identifier for this BrandMaker instance
`events`	JSON array	Array of the transactions submitted in this request
`module`	String	Module used in webhook registration.
`operation`	String	Operation specified in webhook registration. (Module specific), like i.e. "UPDATE" or "DEPUBLISH"
`entity`	String	Entity type used in webhook registration. (Module specific), like i.e. "job" or "invoice-node"
`timestamp`	Long	the Unix timestamp indicating when the event has been created, seconds since the epoch.
`objectId`	JSON Object	The composite ID of the entity for which the event was emitted.
`data`	JSON Object	The effective payload of this event (optional). See chapter 7.
`userid`	String	A user ID if applicable (optional)

5.4 Notes on the Processing of Transaction Messages

Since it is not unlikely that a high number of requests will be sent against the Consumer in a short period of time, it is necessary that the consumer has a correspondingly good response time behavior. For this reason, the actual processing of the messages should not take place immediately after receipt, but they should first be queued in an internal message queue.

There they can then be picked up in the second step and processed asynchronously. This prevents the consumer from blocking and losing messages. MBI does attempt to redeliver them, however the Consumer is automatically disabled if too many errors occur in a row. The respective timeouts can be set in the registry, but again we advise against too large values (> 10 seconds).

The example Consumer described in the next chapter uses Apache ActiveMQ, for the internal queue, you can equally use another message broker of your choice.

6 Example Implementation of a Consumer

To facilitate the creation of MBI Consumers and to illustrate the basic principle, we have provided a sample implementation at https://github.com/brandmaker/MBI-Consumer.

Some basic recommendations, as well as how it works, are explained below.

6.1 Basic Structure

The Consumer should process the following processing steps:

1. Receive and the request including the request body
2. Validation of the request, signature verification
3. Deserialization of the event array
4. For each event in the array
    a. Validation of the event
    b. Set event to an internal processing queue. We recommend not to process the events synchronously, because this can lead to a blocking of the Consumer in case of long runtimes
5. Return response receipt to MBI service

The internal processing queue receives its own Consumer (“listener”), which processes the set events in sequence. This is where the actual processing of the data takes place.

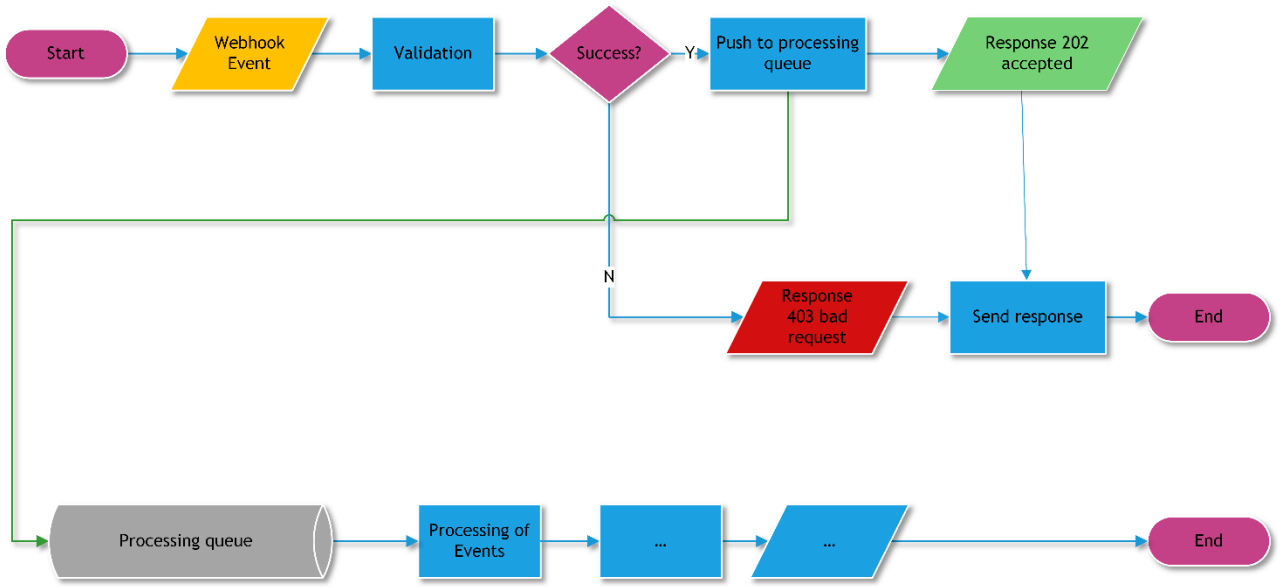


Figure 2 - Program flow Consumer

6.2 General Notes

The following aspects must be considered during implementation:

- The MBI may send a series of events in a single POST request. Thus, it is not possible to specifically return the success or failure of processing a single event to the MBI. In principle, it is possible to set the “batch size” to “1” during registration, but this is not recommended, since a high number of transactions will increase the system load both on the BrandMaker system and in the Consumer.
- Events can be deduplicated in the MBI. A sequence of e.g. “Element Updated” events are thus combined into one event and only the last event is sent. Thus, an event is not always sent for every effective transaction. Deduplication is performed based on the following criteria:
    - They must contain the same transaction category (e.g. “UPDATE”)
    - They must concern the same object (e.g. Job Manager Task “42”)
    - Transactions identified in this way must be in immediate succession

  Whether events are deduplicated or all events are sent depends on the module. Refer to chapter 7 below in the documentation of the events for each module.
- If a Consumer returns too many errors in a row, it is automatically disabled. In doing so, the MBI registers an error when
    - The Consumer returns a different http status than `200 – ok` or `202 – accepted`
    - One of the timeouts expires, which can be set in the administration of the consumer registration.
        - Connection Timeout – Time until a TCP/IP connection is established.
        - Response Timeout – Time until a response is received from the Consumer.

  The MBI will first attempt to redeliver the messages. If this also fails, the Consumer is deactivated and no longer receives messages.

6.3 Signature Validation

The validation of the signature in the Consumer ensures that the received request is unaltered and was actually sent by the customer’s BrandMaker system.

According to https://tools.ietf.org/id/draft-cavage-http-signatures-07.html a list of headers is sent for this purpose, the values of which are used to form the message to be signed. For MBI this includes the headers `date`, `method` und `host`. If the Consumer is operated behind a reverse proxy, the host header may not contain the value used for signing, but the IP or the name of the effective

server. In this case, make sure that the reverse proxy includes the header `x-forwarded-server` in the request and use its value in the signature validation. For more information, please refer to the published example on GitHub.

6.4 Using the Sample Consumer on GitHub

The sample consumer provided at https://github.com/brandmaker/MBI-Consumer can be used by interested developers. It is not subject to any legal restriction (Apache Commons License). BrandMaker does not warrant that the published source code is error-free, nor that it can meet any user's requirements.

7 Documentation of Available Transactions per Module

7.1 Job Manager

> **Note**
>
> The provided payload data may vary. It depends on which fields are configured in the respective job template.

Events in the Job Manager are deduplicated based on the “Category” column.

MBI Event Name	Category	Description	Provided Payload Data

JM.JOB_CREATE	CREATE	A new job has been created	{
			"systemBaseUri": "https://is-int-mrm.brandmaker.com",
			"systemId": "610-067-857",
			"customerId": "kfb-kzk-nbn",
			"namespace": null,
			"events": [
			{
			"module": "JM",
			"operation": "JM.CREATE",
			"entity": "Job",
			"timestamp": 2903978565383543,
			"objectId": {
			"L10N_LOCALE_ID": 0,
			"ORDINAL_NUMBER": 17124,
			"INSTANCE_ID": 17239
			},
			"data": {
			"parentOrdinalNumber": null,
			"sourceOfChange": "Gui",
			"newWorkflowStatus": "CREATED",
			"values": {
			"workflow__object__id": 29507,
			"last__modification__date": "2022-01-25T11:08:43",
			"creator__": {
			"name": "Admin User",
			"id": 3821,
			"login": "admin"
			},
			"workflow__overdue__date": "",
			"job__id": 17124,
			"job__state": "ACTIVE",
			"job__type__pseudo__variable": 138979,
			"job__name": "admin type",
			"deadline__": "",
			"owner__": {
			"groupId": "-1",
			"userName": "Admin User",
			"userId": "3821"
			},
			"current__step__overdue__date": "",
			"themes__": [],
			"default__media": [],
			"job__id__formatted": "17124",
			"workflow__timing": {
			"duration": null,
			"dueDate": null,
			"startDate": null
			},
			"current__step__start__date": "",
			"ob__owner__": {
			"id": null
			},
			"workflow__start__date": "",
			"create__date": "2022-01-25T11:08:43"
			},
			"metaType": "Job",
			"eventName": "DSE_OBJECT_CHANGE_WF_STATUS_JMS_SUBJECT",

MBI Event Name	Category	Description	Provided Payload Data

			<pre>"typeId": 138979
			},
			"userId": "admin"
			}
			]
			}</pre>

JM.JOB_UPDATE	UPDATE	A job has been modified / changed	<pre>{
			"systemBaseUri": "https://is-int-mrm.brandmaker.com",
			"systemId": "610-067-857",
			"customerId": "kfb-kzk-nbn",
			"namespace": null,
			"events": [
			{
			"module": "JM",
			"operation": "JM.UPDATE",
			"entity": "Job",
			"timestamp": 2904075002081313,
			"objectId": {
			"L10N_LOCALE_ID": 0,
			"ORDINAL_NUMBER": 17124,
			"INSTANCE_ID": 1723
			},
			"data": {
			"changedValues": {"job__name":"old job name"},
			"parentOrdinalNumber": null,
			"sourceOfChange": "Gui",
			"values":
			{"workflow__object__id":29507,"last__modification__date":"2022-01-25T11:10:20","creator__":{"name":"Admin User","id":3821,"login":"admin"},"workflow__overdue__date":"","job__id":17124,"job__state":"ACTIVE","job__type__pseudo__variable":138979,"job__name":"new job name","deadline__":"","owner__":{"groupId":"-1","userName":"Admin User","userId":"3821"},"current__step__overdue__date":"","themes__":[],"default__media":[],"job__id__formatted":"17124","workflow__timing":{"duration":null,"dueDate":null,"startDate":null},"current__step__start__date":"","ob__owner__":{"id":null},"workflow__start__date":"","create__date":"2022-01-25T11:08:43"},
			"metaType": "Job",
			"eventName": "DSE_OBJECT_CHANGE_VALUES_JMS_SUBJECT",
			"typeId": 138979
			},
			"userId": "admin"
			}
			]
			}</pre>

JM.JOB_DELETE	DELETE	Delete a job	<pre>{
			"systemBaseUri": "https://is-int-mrm.brandmaker.com",
			"systemId": "610-067-857",
			"customerId": "kfb-kzk-nbn",
			"namespace": null,
			"events": [
			{
			"module": "JM",
			"operation": "JM.JOB_DELETE",
			"entity": "Job",
			"timestamp": 2904238056841610,
			"objectId": {
			"L10N_LOCALE_ID": 0,
			"ORDINAL_NUMBER": 17124,
			"INSTANCE_ID": 17239
			},
			"data": {
			"parentOrdinalNumber": null,
			"sourceOfChange": "Gui",
			"newWorkflowStatus": "DELETED",
			"values": {
			"workflow__object__id": 29508,
			"last__modification__date": "2022-01-25T11:13:03",
			"creator__": {
			"name": "Admin User",
			"id": 3821,
			"login": "admin"
			},
			"workflow__overdue__date": "",
			"job__id": 17124,
			"job__state": "DELETED",
			"job__type__pseudo__variable": 138979,
			"job__name": "new job name",
			"deadline__": "",</pre>

MBI Event Name	Category	Description	Provided Payload Data

			"owner__": {
			"groupId": "-1",
			"userName": "Admin User",
			"userId": "3821"
			},
			"current__step__overdue__date": "",
			"themes__": [],
			"default__media": [],
			"job__id__formatted": "17124",
			"workflow__timing": {
			"duration": null,
			"dueDate": null,
			"startDate": "2022-01-25"
			},
			"current__step__start__date": "01/25/2022",
			"ob__owner__": {
			"id": null
			},
			"workflow__start__date": "",
			"create__date": "2022-01-25T11:08:43"
			},
			"metaType": "Job",
			"eventName": "DSE_OBJECT_CHANGE_WF_STATUS_JMS_SUBJECT",
			"typeId": 138979,
			"oldWorkflowStatus": "ACTIVE"
			},
			"userId": "admin"
			}
			]
			}

JM.JOB_STEP_CHANGED	UPDATE	A job has been forwarded.
		A job has been sent back.
			{
			"systemBaseUri": "https://is-int-mrm.brandmaker.com",
			"systemId": "610-067-857",
			"customerId": "kfb-kzk-nbn",
			"namespace": null,
			"events": [
			{
			"module": "JM",
			"operation": "JM.JOB_STEP_CHANGED",
			"entity": "Job",
			"timestamp": 2904155089760823,
			"objectId": {
			"L10N_LOCALE_ID": 0,
			"ORDINAL_NUMBER": 17124,
			"INSTANCE_ID": 17239
			},
			"data": {
			"values": {
			"workflow__object__id": 29507,
			"last__modification__date": "2022-01-25T11:11:40",
			"creator__": {
			"name": "Admin User",
			"id": 3821,
			"login": "admin"
			},
			"workflow__overdue__date": "",
			"job__id": 17124,
			"job__state": "ACTIVE",
			"job__type__pseudo__variable": 138979,
			"job__name": "new job name",
			"deadline__": "",
			"owner__": {
			"groupId": "1",
			"userName": "Admin User",
			"userId": "3821"
			},
			"current__step__overdue__date": "",
			"themes__": [],
			"default__media": [],
			"job__id__formatted": "17124",
			"workflow__timing": {
			"duration": null,
			"dueDate": null,
			"startDate": "2022-01-25"
			},
			"current__step__start__date": "01/25/2022",
			"ob__owner__": {
			"id": null
			},
			"workflow__start__date": "",

MBI Event Name	Category	Description	Provided Payload Data

			<pre>"create__date": "2022-01-25T11:08:43"
			},
			"newWorkflowStepName": "step1",
			"workflowName": "@qa 2steps",
			"oldWorkflowStepId": "660",
			"newWorkflowStepId": "661",
			"parentOrdinalNumber": "null",
			"newWorkflowStepNumber": "1",
			"oldWorkflowStepName": "step0",
			"sourceOfChange": "Gui",
			"metaType": "Job",
			"eventName": "DSE_OBJECT_CHANGE_WF_STEP_JMS_SUBJECT",
			"typeId": 138979,
			"oldWorkflowStepNumber": 0
			},
			"userId": "admin"
			}
			]
			}</pre>

JM.JOB_FINISH	UPDATE	A job has ended	<pre>{
			"systemBaseUri": "https://is-int-mrm.brandmaker.com",
			"systemId": "610-067-857",
			"customerId": "kfb-kzk-nbn",
			"namespace": null,
			"events": [
			{
			"module": "JM",
			"operation": "JM.JOB_FINISH",
			"entity": "Job",
			"timestamp": 2904304431000208,
			"objectId": {
			"L10N_LOCALE_ID": 0,
			"ORDINAL_NUMBER": 17124,
			"INSTANCE_ID": 17239
			},
			"data": {
			"parentOrdinalNumber": null,
			"sourceOfChange": "Gui",
			"newWorkflowStatus": "FINISHED",
			"values": {
			"workflow__object__id": 29508,
			"last__modification__date": "2022-01-25T11:14:10",
			"creator__": {
			"name": "Admin User",
			"id": 3821,
			"login": "admin"
			},
			"workflow__overdue__date": "",
			"job__id": 17124,
			"job__state": "FINISHED",
			"job__type__pseudo__variable": 138979,
			"job__name": "new job name",
			"deadline__": "",
			"owner__": {
			"groupId": "1",
			"userName": "Admin User",
			"userId": "3821"
			},
			"current__step__overdue__date": "",
			"themes__": [],
			"default__media": [],
			"job__id__formatted": "17124",
			"workflow__timing": {
			"duration": null,
			"dueDate": null,
			"startDate": "2022-01-25"
			},
			"current__step__start__date": "01/25/2022",
			"ob__owner__": {
			"id": null
			},
			"workflow__start__date": "",
			"create__date": "2022-01-25T11:08:43"
			},
			"metaType": "Job",
			"eventName": "DSE_OBJECT_CHANGE_WF_STATUS_JMS_SUBJECT",
			"typeId": 138979,
			"oldWorkflowStatus": "ACTIVE"
			},
			"userId": "admin"</pre>

MBI Event Name	Category	Description	Provided Payload Data
			<pre> }] }</pre>

JM.JOB_CANCEL	UPDATE	A running job is cancelled	<pre>{
			    "systemBaseUri": "https://is-int-mrm.brandmaker.com",
			    "systemId": "610-067-857",
			    "customerId": "kfb-kzk-nbn",
			    "namespace": null,
			    "events": [
			        {
			            "module": "JM",
			            "operation": "JM.JOB_CANCEL",
			            "entity": "Job",
			            "timestamp": 2904181333269775,
			            "objectId": {
			                "L10N_LOCALE_ID": 0,
			                "ORDINAL_NUMBER": 17124,
			                "INSTANCE_ID": 17239
			            },
			            "data": {
			                "parentOrdinalNumber": null,
			                "sourceOfChange": "Gui",
			                "newWorkflowStatus": "CANCELED",
			                "values": {
			                    "workflow__object__id": 29507,
			                    "last__modification__date": "2022-01-25T11:12:06",
			                    "creator__": {
			                        "name": "Admin User",
			                        "id": 3821,
			                        "login": "admin"
			                    },
			                    "workflow__overdue__date": "",
			                    "job__id": 17124,
			                    "job__state": "CANCELED",
			                    "job__type__pseudo__variable": 138979,
			                    "job__name": "new job name",
			                    "deadline__": "",
			                    "owner__": {
			                        "groupId": "1",
			                        "userName": "Admin User",
			                        "userId": "3821"
			                    },
			                    "current__step__overdue__date": "",
			                    "themes__": [],
			                    "default__media": [],
			                    "job__id__formatted": "17124",
			                    "workflow__timing": {
			                        "duration": null,
			                        "dueDate": null,
			                        "startDate": "2022-01-25"
			                    },
			                    "current__step__start__date": "01/25/2022",
			                    "ob__owner__": {
			                        "id": null
			                    },
			                    "workflow__start__date": "",
			                    "create__date": "2022-01-25T11:08:43"
			                },
			                "metaType": "Job",
			                "eventName": "DSE_OBJECT_CHANGE_WF_STATUS_JMS_SUBJECT",
			                "typeId": 138979,
			                "oldWorkflowStatus": "ACTIVE"
			            },
			            "userId": "admin"
			        }
			    ]
			}</pre>

JM.INITIAL_LOAD	SYNC	Sync all existing jobs in their current state	JSON which contains all data of all jobs, reflecting the current state

7.2 Marketing Planer

The Marketing Planer will be integrated into the MBI in one of the next releases of BrandMaker.

7.3 Review Manager

The Review Manager will be integrated into the MBI in one of the next releases of BrandMaker.

7.4 Resource Management

Resource Management will be integrated into MBI in one of the next releases of BrandMaker.

8 Security Advice

8.1 SSL Encryption

The MBI only processes URLs that use SSL-encrypted HTTP.

The Consumer operator must ensure that a valid SSL certificate is installed on its system. This must be signed by a known root CA.

Self-signed certificates are not accepted and lead to a connection error and consequently to a deactivation of the Consumer in the MBI.

8.2 Request Validation

All requests against the Consumer are signed by the MBI according to https://tools.ietf.org/id/draft-cavage-http-signatures-07.html The creator of the Consumer is responsible for validating requests against the signature and corresponding public key of the BrandMaker system before processing them and discarding invalid requests.

8.3 Requests to the BrandMaker API

An MBI Consumer is not “per se” authorized to send requests to the BrandMaker APIs. Rather, this requires valid authentication to the BrandMaker system. BrandMaker provides the OAuth2-compliant service “BrandMaker CAS” for this purpose.
See: https://developers.brandmaker.com/guides/auth/

Please note that all requests to the BrandMaker system APIs are always made in the context of an authenticated and authorized user. Its rights and roles affect the result of each request.