BrandMaker

Message Based Integration Service

Handbuch

Version 7.3

DE

25. März 2022

Copyright

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

Ihr Feedback ist uns wichtig!

Für Hinweise auf Fehler sind wir jederzeit dankbar. Senden Sie uns einfach eine E-Mail an documentation@brandmaker.com.

1 Einführung

Mit Release 7.3 stellt BrandMaker erstmals den „Message Based Integration Service“ bereit. Dieser zentrale Service dient der kundenseitigen Integration des BrandMaker-Systems in die Infrastruktur des Kunden.

Der Dienst basiert auf dem sogenannten „Publish / Subscriber“-Prinzip. Beginnend mit dem Modul Job Manager in Release 7.3, senden alle Module in BrandMaker Informationen über Benutzertransaktionen im System an den MBI. „Abonnenten“ („Subscriber“) des Dienstes können diese Transaktionsnachrichten empfangen und verarbeiten.

Auf diese Weise ist eine lose Koppelung des BrandMaker-Systems und seiner Module mit Fremdanwendungen möglich. Weiterhin stehen den Fremdanwendungen die umfangreichen APIs des BrandMaker-Systems und seiner Module zur Verfügung, um weitere Daten zu lesen oder zu modifizieren.

1.1 Terminologie

Verweise innerhalb der Tabelle sind als anklickbare Links mit einem → dargestellt.

Consumer	→Subscriber
Event	Die an den Consumer übermittelte →Transaktionsnachricht enthält eine Liste von „Events“ – im System aufgetretenen Ereignissen sowie die dazugehörigen Daten (→Payload)
MBI	BrandMaker **M**essage **B**ased **I**ntegration Service, bezeichnet den Dienst als Ganzes.
Payload	Nutzdaten der Nachricht. Die Payload ist Modul- und Eventabhängig
Producer	→Publisher
Publisher	Sender einer Transaktion, im Allgemeinen ein Modul des BrandMaker-Systems wie z.B. der Job Manager oder der Marketing Planer
Subscriber	Ein am MBI registriertes System zum Empfang einer Transaktionsnachricht
Transaktion	Vom Benutzer ausgelöstes oder auch automatisches Ereignis innerhalb des BrandMaker-Moduls, z.B. „ein neuer Task im Job Manager wurde angelegt“
Transaktionsnachricht	Information über eine (stattgefundene) Transaktion, die der MBI an registrierte Subscriber oder Consumer sendet
Webhook	Generell ein Verfahren zur asynchronen Datenintegration zwischen einzelnen, abgegrenzten Softwaresystemen. Im Speziellen der Dienst, der eine solche Funktion bereitstellt: ein →Consumer registriert sich an einem Webhook, um von diesem Daten zu empfangen.

1.2 Weiterführende Informationen

Gute Einführungen und Hintergrundwissen zum Thema „Publish / Subscriber“ oder „Event Driven Architecture“ finden Sie in dieser Linksammlung:

- 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 Aufbau dieses Handbuchs

Dieses Handbuch beschreibt die verschiedenen Aspekte von MBI.

- Architekturübersicht in Kapitel 3
- Administration der Consumer-Registrierungen im BrandMaker-System in Kapitel 4
- Anforderungen an die Consumer-API im Kapitel 2
- Beispiel-Implementierung eines Consumers in Kapitel 6
- Dokumentation der Transaktionen und bereitgestellten Daten der Module ab Kapitel 6.2
    a. Job Manager
    b. Marketing Planer
    c. Review Manager
    d. Resource Management

2 Randbedingungen

2.1 Administration

Der Leser muss grundsätzlich mit der Administration des BrandMaker-Systems vertraut sein. Dieses Manual beschreibt die spezifischen Einstellungen der Consumer-Registrierung. Sie benötigen Administrator-Zugang zu Ihrem BrandMaker-System, um die notwendigen Einstellungen vornehmen zu können.

Vorbedingungen

1. Navigieren Sie auf die *Seite > Administration > Systemeinstellungen*.
2. Suchen Sie nach dem Element `brandmaker.mbi.url`
3. Prüfen Sie eine bestehende URL. Andernfalls tragen Sie die noch fehlende URL in diesem Feld ein. Die korrekte URL erhalten sie über den BrandMaker-Support.

2.2 Entwicklung eines Consumers

Es wird vorausgesetzt, dass der Leser, resp. der Entwickler auf Kundenseite mit Technologien wie HTTP, REST und der Programmierung von Webdiensten vertraut ist.

Aufgeführte Beispiel-Listings sind exemplarisch in Java erstellt, es besteht aber kein Zwang und keine Präferenz für irgendeine Programmiersprache.

Weiterhin ist die Kenntnis der Anwendung und der Konfiguration des BrandMaker-Systems erforderlich. Der Entwickler benötigt Administrator-Zugang zum BrandMaker-System, um die notwendigen Einstellungen vornehmen zu können.

Um Daten mithilfe eines Drittsystems aus dem BrandMaker-System automatisiert abzuholen und zu speichern, ist die Kenntnis der entsprechenden Entwicklungsumgebungen und APIs des Drittsystems notwendig. Diese sind ebenfalls nicht Gegenstand dieser Dokumentation. Das Drittsystem wird in diesem Zusammenhang immer als Blackbox dargestellt.

2.3 Kontextabgrenzung

Die angebotenen Schnittstellen dienen der Nahe-Echtzeit-Synchronisation der Datenbestände im BrandMaker-System mit Drittanwendungen. Andere Einsatzzwecke werden nicht unterstützt.

3 Architekturübersicht

Die Grundarchitektur des BrandMaker MBI sieht wie folgt aus:

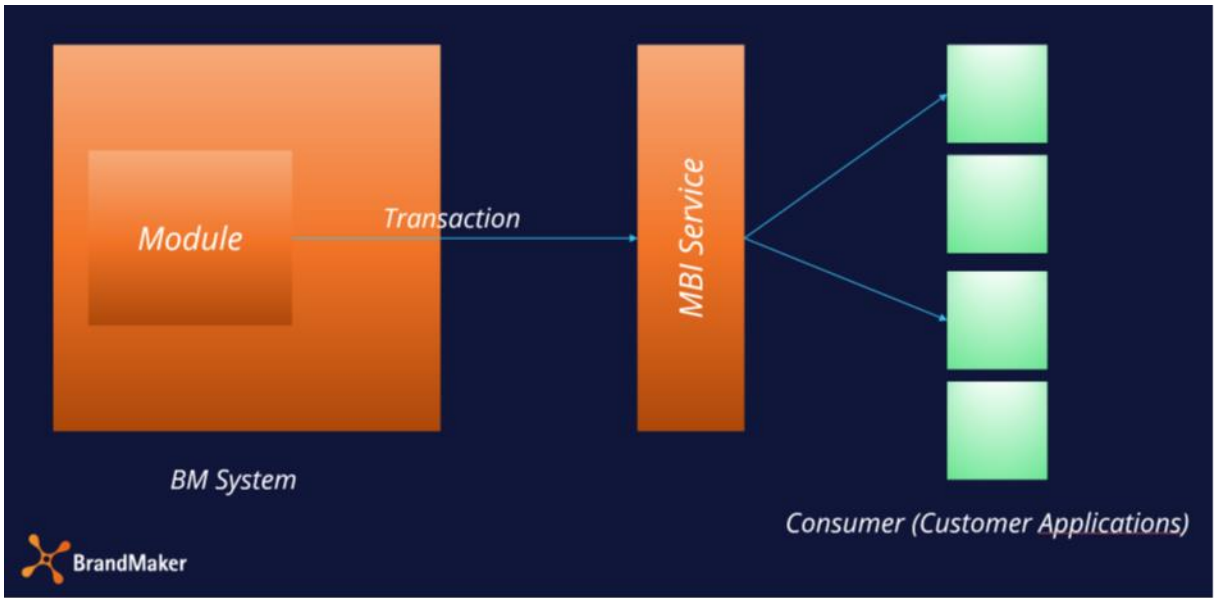


Abbildung 1 – Prinzip-Skizze BrandMaker MBI

Das Modul sendet Informationen über abgeschlossene Transaktionen an den MBI. Der MBI prüft, ob für das Modul und die jeweilige Transaktion eine Registrierung vorliegt und sendet die Information sowie eine Reihe von Zusatzangaben über das Quellsystem und -modul an die registrierten Fremdanwendungen.

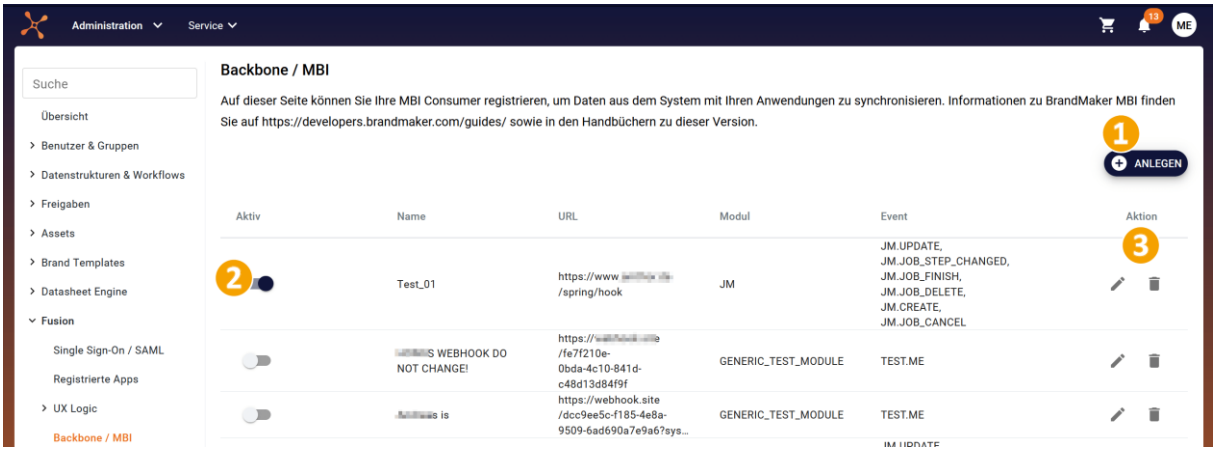
Die Fremdanwendungen müssen dazu eine REST-API bereitstellen, die in der Lage ist, die zugesendeten Daten anzunehmen und über den Empfang eine Quittung zurückzusenden.

Die Anforderungen an diese bereitzustellende REST-API sind im weiteren Verlauf diese Manuals in Kapitel 2 beschrieben. Generell sendet der MBI per HTTP-POST-Anfrage die Transaktionsdaten im JSON-Format an den registrierten Consumer.

Die Registrierung eines Consumers findet im jeweiligen Kundensystem in der Administration statt.

4 Administration: Consumer-Registrierung für MBI

Loggen Sie sich als Administrator auf Ihrem BrandMaker-System ein, um die erforderlichen Einstellungen vorzunehmen. Sie erreichen die Registrierung unter > *Administration* > *Fusion* > *Backbone / MBI*:



Ihre Bildschirmansicht sollte jetzt der obigen ähneln. Einige Bildschirmtexte können bis zur Release-Version noch Änderungen unterliegen.

1 Anlegen

Registrieren Sie den MBI Consumer, siehe 4.1.

3 Aktion-Buttons

In diesem Bereich erreichen Sie folgende Aktionen:

- MBI Consumer bearbeiten

  Mit Klick auf das Stiftsymbol können Sie Änderungen an der Konfiguration vornehmen.

- MBI Consumer löschen

  Nach Klick auf das Papierkorb-Symbol, öffnet sich eine Sicherheitsabfrage, bevor Sie den Consumer endgültig löschen können.

2 Slider „Active“

Verwenden Sie diesen Slider, um einen MBI Consumer zu aktivieren oder zu deaktivieren.

Eigenschaften eines Consumers

Eingabe	Beschreibung
Event Timeout	Wie lange wird auf Antwort des Consumers gewartet? [Einheit s = Sekunden]; Beispiel: 30. Nach Ablauf des Timeouts wird ein Fehler registriert. Sollten zu viele Fehler in Folge auftreten, wird der Consumer deaktiviert. Siehe dazu 5.4.

Eingabe	Beschreibung
Verzögerung bei Wiederholung	[Einheit s = Sekunden]; Beispiel: 60. MBI wartet so lange, wie hier angegeben. Danach wird ein erneuter Versuch unternommen, die Nachricht zuzustellen.
Max Retries	Maximale Anzahl von Versuchen, nach Timeout Daten zu senden.
Max Events	Höchstwert der Events, die auf einmal übermittelt werden. Diese werden in einem Batch und nicht einzeln versendet.
Verpasste Nachrichten senden	Fehlende Transaktions-Nachrichten, die nicht übermittelt werden konnten, werden nachgereicht, sobald der Consumer wieder verfügbar ist. Bei Inaktivität werden sie daher zwischengespeichert.
Modul und Events	Hier wählen Sie aus, von welchen Modulen und Events Daten empfangen werden sollen.

Hinweis

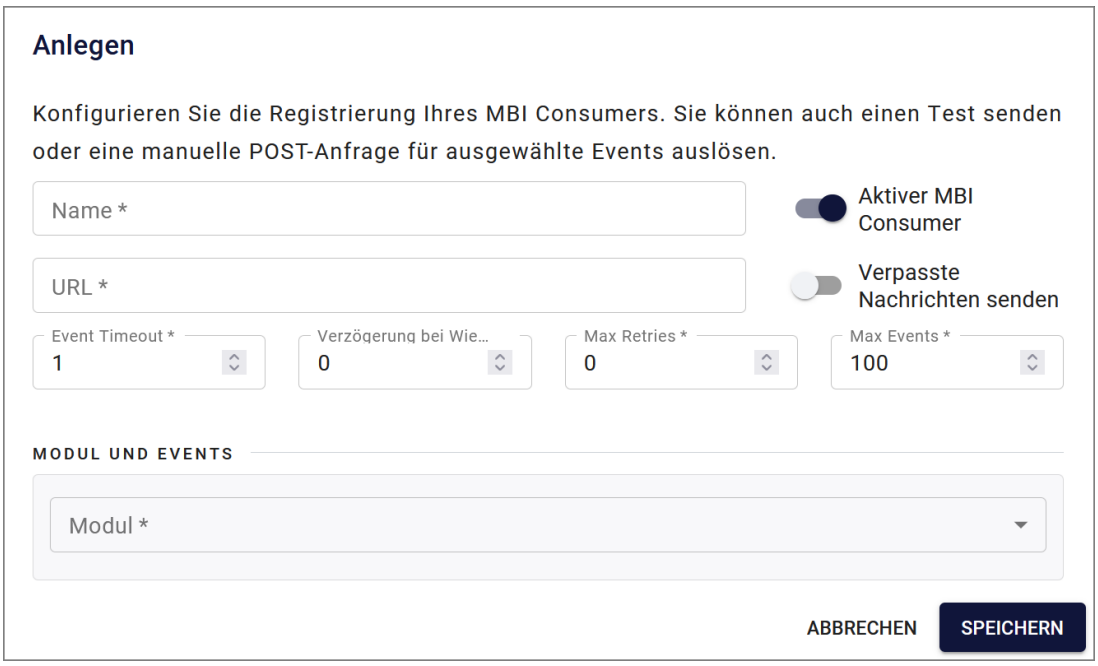
Die Höchstgrenze der erlaubten Eingabe beträgt für:

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

4.1 Consumer anlegen

Klicken Sie *Create*.

Der folgende Dialog wird angezeigt.

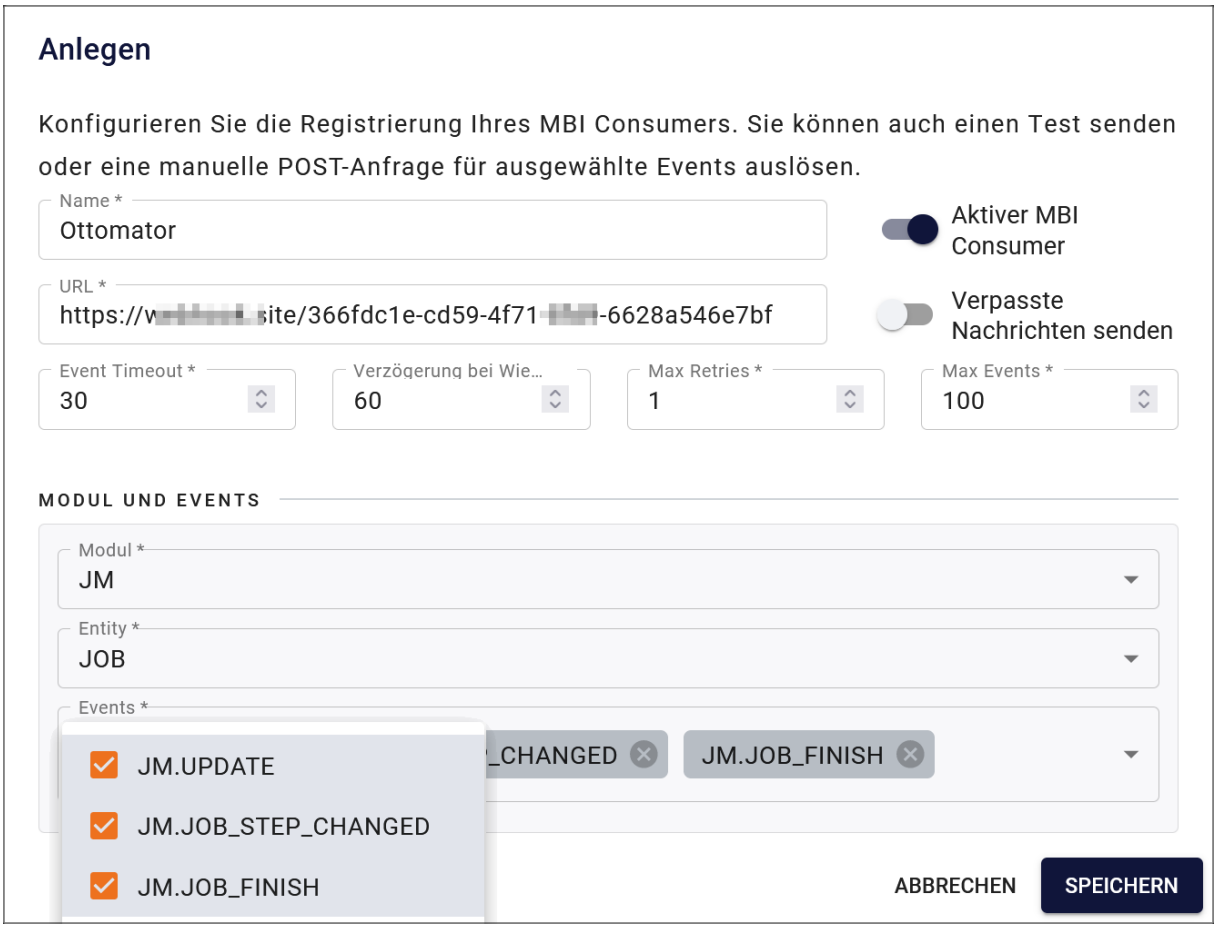


Bearbeiten Sie die Felder im oberen Bereich. Pflichtfelder sind mit einem Stern (*) markiert. Beachten Sie für eine Beschreibung der Felder das Kapitel 4.

Hinweis: Die Consumer-URL muss ordnungsgemäß für das HTTPS-Protokoll registriert und konfiguriert sein. Experimentelle Testumgebungen mit *localhost*-Konstrukten funktionieren nicht.

Der Slider *Aktiver MBI Consumer* ist in der Voreinstellung aktiviert. Wenn Sie dies nicht wünschen, weil die Umgebung noch nicht zu Ende konfiguriert ist, oder Sie den Consumer noch nicht freischalten möchten, speichern Sie die Konfiguration für den neuen Consumer erst dann, nachdem Sie den Slider deaktiviert haben.

Legen Sie im Bereich *Modul und Events* in der Auswahllisten *Module*, *Entity* und *Events* fest, auf welche Ereignisse reagiert werden soll:



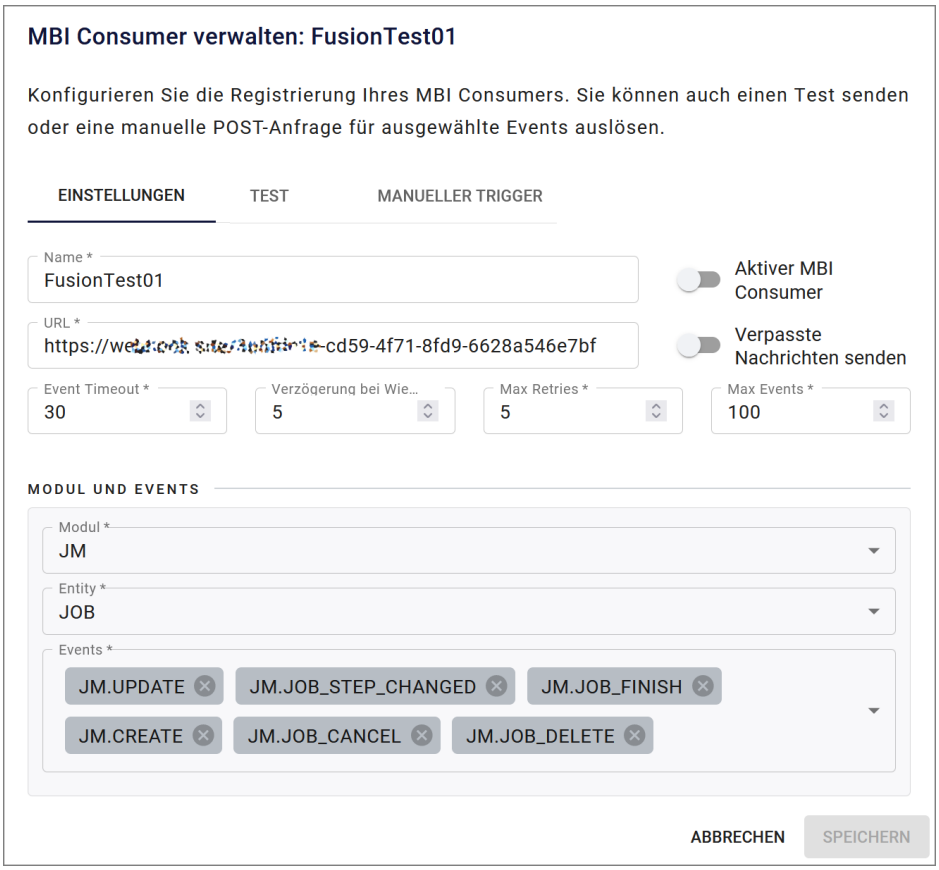
Klicken Sie *Speichern*.

Sie haben einen Consumer registriert.

4.2 MBI Consumer verwalten

1. Klicken Sie in der Zeile des Consumers, den Sie bearbeiten möchten, rechts in der Zeile auf das Stiftsymbol.

Der Dialog *MBI Consumer verwalten* wird angezeigt:



2. Bearbeiten Sie die Eigenschaften des Consumers auf dem Reiter *Einstellungen*.
3. Klicken Sie *Speichern*.

Sie haben den MBI Consumer bearbeitet.

4.3 MBI Consumer testen

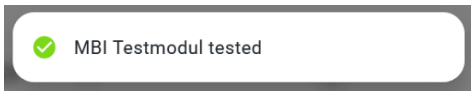
1. Klicken Sie in der Zeile des Consumers, den Sie testen möchten, rechts in der Zeile das Stiftsymbol.

Der Dialog *MBI Consumer verwalten* wird angezeigt:

2. Wechseln Sie auf den Reiter *Test*.

Zum Testen klicken Sie den *Test*-Button.

Es wird eine Anfrage mit zufälligen Werten gesendet. Prüfen Sie, ob der Empfang und die Verarbeitung mit Ihrem Consumer funktionieren. Im Erfolgsfall wird eine Benachrichtigung wie unten abgebildet eingeblendet.

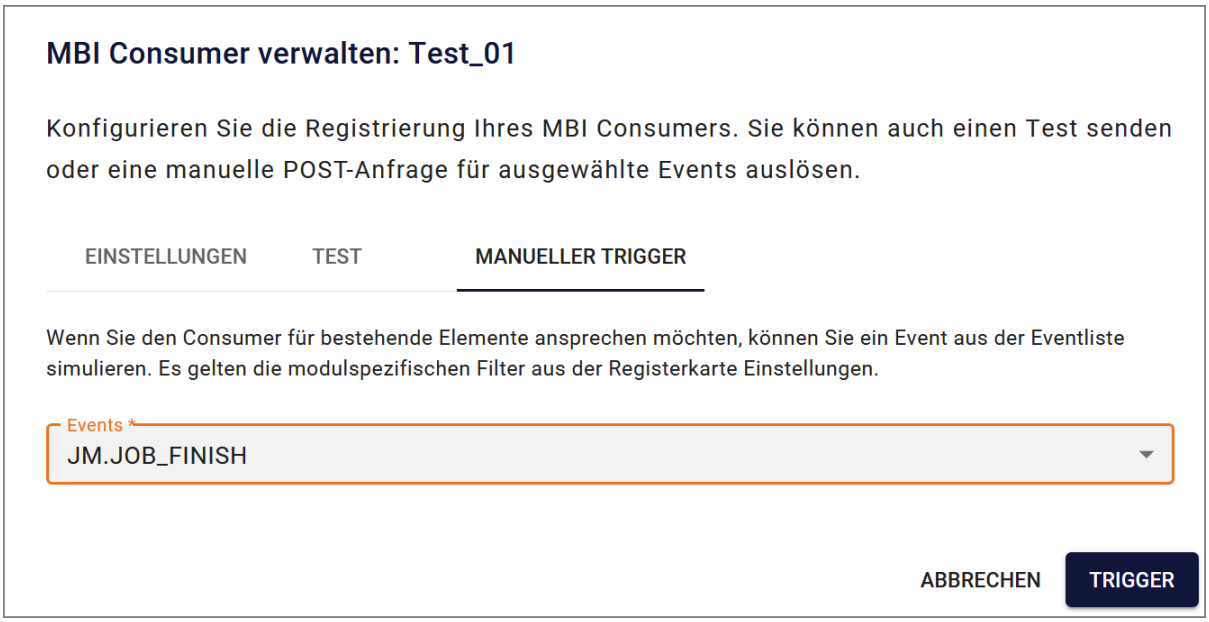


4.4 MBI Consumer manuell auslösen

Wenn Sie den Consumer für bestehende Elemente auslösen möchten, können Sie ein Event aus der Eventliste simulieren.

1. Klicken Sie in der Zeile des Consumers, den Sie testen möchten, rechts das Stiftsymbol.

   Der Dialog *MBI Consumer verwalten* wird angezeigt:



2. Wechseln Sie auf den Reiter *Manueller Trigger*.
3. Wählen Sie ein Event aus dem Auswahlmenü. Es gelten die Filter im Reiter *Einstellungen*.
4. Klicken Sie den Button *Trigger* zum Auslösen.
5. Zu Testzwecken wird eine fingierte Nachricht an den Consumer generiert. Auch hier erhalten Sie Rückmeldung in Form einer eingeblendeten Benachrichtigung.

Sie können das Modul *GENERIC_TEST_MODULE* auswählen, wie der Name bereits andeutet, um generische Tests durchzuführen.

5 Anforderungen an die REST-API des Consumers

Ein MBI-Consumer muss zum Empfang der Transaktionsnachrichten eine REST-Schnittstelle bereitstellen, deren URL in der Registrierung hinterlegt wird.

Hinweis

Bitte beachten Sie, dass diese URL über das öffentliche Internet zugänglich sein muss. Darüber hinaus akzeptiert MBI nur SSL-verschlüsselte Verbindungen („https“).

5.1 Implementierung der REST-Schnittstelle

Der MBI sendet eine HTTP-POST-Anfrage mit den weiter unten beschriebenen Daten an den Consumer. Dieser muss daher in der Lage sein, diese POST-Anfrage zu empfangen und zu verarbeiten.

Eine Anfrage kann dabei eine Reihe von Transaktionen beinhalten (siehe dazu Attribut `events` in der weiter unten beschriebenen Datenstruktur). Die maximale Anzahl der in einem Batch übermittelten Nachrichten wird bei der Registrierung des Consumers in der Administration des BrandMaker-Systems hinterlegt.

Wir empfehlen, nur die POST-Methode zu implementieren und alle anderen HTTP-Methoden mit HTTP-Status 405 („Method not allowed“) zurückzuweisen.

Zur Veranschaulichung kann der Beispiel-Consumer verwendet werden, der auf GitHub veröffentlicht ist: https://github.com/brandmaker/MBI-Consumer.

5.2 Validierung der Anfrage im Consumer

Da es sich um eine asynchrone Batchverarbeitung handelt, muss sichergestellt ein, dass die beim Consumer eintreffende Anfrage auch von der BrandMaker-Instanz gesendet wurde und unverfälschte und korrekte Daten enthält.

Aus diesem Grunde wird eine Signatur im Anfrage-Header mitgesendet, mithilfe derer die Authentizität der Anfrage im Consumer validiert werden kann. Bei der Anfrage handelt es sich um eine gemäß https://tools.ietf.org/id/draft-cavage-http-signatures-07.html signierte HTTP-Nachricht.

Die Signatur umfasst den gesamten Request-Body wodurch die Integrität der Anfrage insgesamt gewährleistet wird. Die Validierung der Nachricht erfolgt im Weiteren dann gemäß dem in obigem RFC-Link beschriebenen Standard und wird hier nicht weiter ausgeführt.

Zur Validierung der Signatur ist der Public-Key des BrandMaker-Systems erforderlich, der über die URL `https://<brandmaker system>/rest/sso/keys/public` bzw. `…/public-for-file` abgeholt werden kann.

Empfehlung für die Praxis

Wir raten Ihnen, diesen Key einmalig abzufragen und statisch in der Anwendung (Consumer) zu hinterlegen. Der Key wird von BrandMaker einmalig generiert und ändert sich für bestehende Systeme nicht.

Wir raten dringend davon ab den Key im Consumer dynamisch über die `systemBaseUri` abzuholen. Dies würde es einem Angreifer ermöglichen, sowohl Public- als auch Private-Key mit einer gefälschten Signatur bereitzustellen und auf diese Weise verfälschte Anfragen zu übermitteln!

5.3 Datenformat der Transaktionsnachricht

Die Transaktionsnachrichten haben ein standardisiertes Datenformat. Einige der Attribute im zugesendeten Objekt sind allerdings vom Kontext des jeweiligen Modules abhängig. Dies sind die gelb markierten Attribute in der nachfolgenden JSON-Struktur.

Der grundlegende Aufbau der Nachricht ist wie folgt:

```
{
  "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 Beschreibung der Attribute

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). Siehe Kapitel 7.
`userid`	String	A user ID if applicable (optional)

5.4 Hinweise zur Verarbeitung der Transaktionsnachrichten

Da es nicht unwahrscheinlich sein wird, das in kurzer Zeit eine hohe Anzahl von Anfragen gegen den Consumer gesendet wird, ist es erforderlich, dass dieser über ein entsprechend gutes Antwortzeitverhalten verfügt. Aus diesem Grunde sollte die eigentliche Verarbeitung der Nachrichten nicht unmittelbar nach Empfang stattfinden, sondern diese zunächst wiederum in eine interne Message Queue eingereiht werden.

Dort können sie dann im zweiten Schritt abgeholt und asynchron verarbeitet werden. Auf diese Weise wird verhindert, dass der Consumer blockiert und Nachrichten verloren gehen. MBI versucht zwar, diese erneut zuzustellen, allerdings wird der Consumer automatisch deaktiviert, wenn zu viele Fehler in Folge stattfinden. Die jeweiligen Timeouts können in der Registrierung eingestellt werden, aber auch hier raten wir von zu großen Werten (> 10 Sekunden) ab.

Der im nächsten Kapitel beschriebene Beispiel-Consumer verwendet für die interne Queue Apache ActiveMQ, Sie können ebenso einen anderen Message Broker Ihrer Wahl einsetzen.

6 Beispiel-Implementierung eines Consumers

Um die Erstellung von MBI-Consumern zu erleichtern und das grundlegende Prinzip zu veranschaulichen, haben wir unter https://github.com/brandmaker/MBI-Consumer eine Beispiel-Implementierung bereitgestellt.

Einige grundlegende Empfehlungen sowie die Funktionsweise werden im Weiteren erläutert.

6.1 Grundlegende Struktur

Der Consumer sollte folgende Verarbeitungsschritte abarbeiten:

1. Empfangen und der Anfrage inklusive des Anfrage-Bodies
2. Validierung der Anfrage, Signatur-Überprüfung
3. Deserialisierung des Event Arrays
4. Für jeden Event im Array
    a. Validierung des Events
    b. Event in eine interne Verarbeitungs-Queue einstellen. Wir empfehlen, die Events nicht synchron zu verarbeiten, da dies bei langen Laufzeiten zu einem Blockieren des Consumers führen kann
5. Antwort-Quittung an den MBI-Service zurücksenden

Die interne Verarbeitungs-Queue erhält einen eigenen Verbraucher („Listener“), der die eingestellten Events der Reihe nach abarbeitet. Hier findet die eigentliche Verarbeitung der Daten statt.

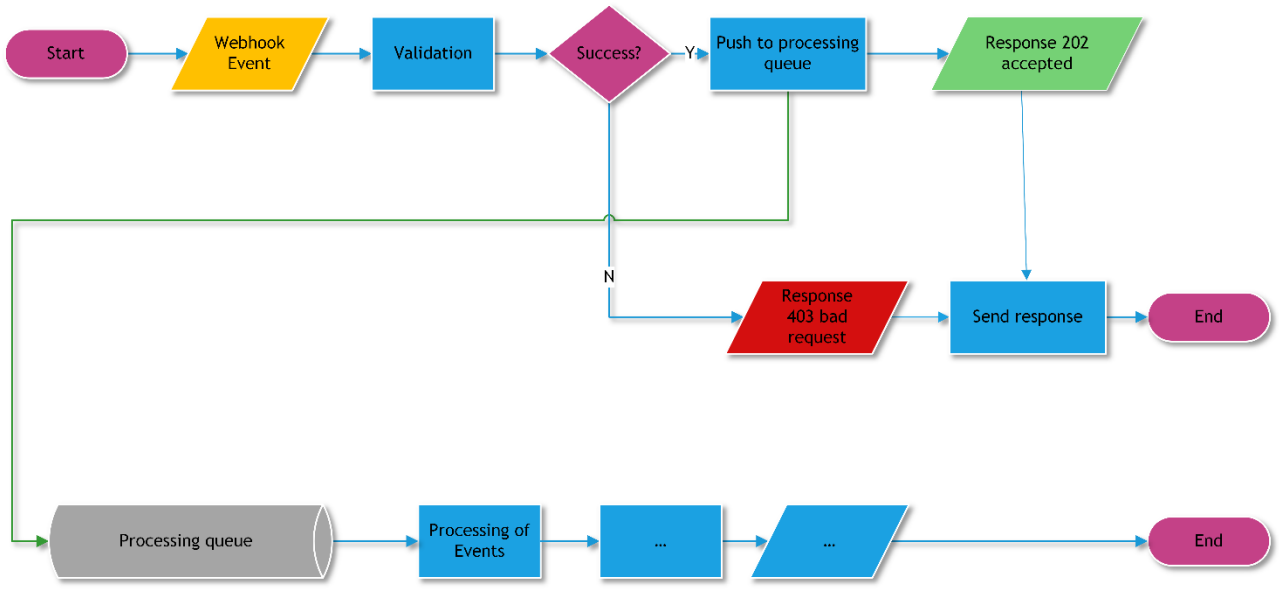


Abbildung 2 - Programmablauf Consumer

6.2 Generelle Hinweise

Folgende Aspekte sind bei der Implementierung zu beachten:

- Der MBI sendet unter Umständen eine Reihe von Events in einer einzelnen POST-Anfrage. Es ist somit nicht möglich, gezielt den Erfolg oder den Misserfolg der Verarbeitung eines einzelnen Events an den MBI zurückzugeben. Prinzipiell kann man die „Batch-Größe“ bei der Registrierung zwar auf „1“ setzen, allerding ist das nicht zu empfehlen, da bei einer hohen Anzahl von Transaktionen die Systemlast sowohl auf dem BrandMaker-System als auch ggf. im Consumer steigen wird.

- Events können im MBI dedupliziert werden. Eine Abfolge von z.B. „Element Updated“-Events werden so zu einem Event zusammengefasst und lediglich das letzte Event gesendet. Somit wird also nicht für jede effektive Transaktion immer ein Event gesendet. Die Deduplizierung erfolgt anhand von folgenden Kriterien:

    - Sie müssen dieselbe Transaktionskategorie beinhalten (z.B. „UPDATE“)
    - Sie müssen das gleiche Objekt betreffen (z.B. Job Manager Task „42“)
    - Diese so identifizierten Transaktionen müssen unmittelbar aufeinanderfolgen

    Ob Events dedupliziert werden oder alle Events gesendet werden, ist modulabhängig. Beachten Sie dazu Kapitel 7 unten in der Dokumentation der Events zu den einzelnen Modulen.

- Wenn ein Consumer in Folge zu viele Fehler zurückmeldet, wird er automatisch deaktiviert. Dabei registriert der MBI einen Fehler, wenn

    - Der Consumer einen anderen http-Status zurückliefert als `200 - ok` oder `202 - accepted`
    - Einer der Timeouts abläuft, die in der Administration der Consumer-Registrierung eingestellt werden können.
        - Connection Timeout – Zeit bis zur Herstellung einer TCP/IP-Verbindung
        - Response Timeout – Zeit bis zum Empfang einer Antwort vom Consumer

    Der MBI wird zunächst versuchen, die Nachrichten erneut zuzustellen. Scheitert auch das, wird der Consumer deaktiviert und erhält keine Nachrichten mehr.

6.3 Signatur-Validierung

Die Validierung der Signatur im Consumer stellt sicher, dass der empfange Request unverfälscht ist und auch tatsächlich vom BrandMaker System des Kunden abgesendet wurde.

Gemäß https://tools.ietf.org/id/draft-cavage-http-signatures-07.html wird dazu eine Liste der Header gesendet, aus deren Werten die zu signierende Nachricht gebildet wird. Diese umfasst beim MBI die

Header `date`, `method` und `host`. Sofern der Consumer hinter einem Reverse-Proxy betrieben wird, kann es sein, dass der Host-Header nicht den Wert enthält, der zur Signierung verwendet wurde, sondern die IP oder den Namen des effektiven Servers. In diesem Falle ist dafür Sorge zu tragen, dass der Reverse-Proxy den Header `x-forwarded-server` in den Request einfügt und es ist dessen Wert in der Signatur-Validierung zu verwenden. Für weitere Informationen verweisen wir auf das veröffentlichte Beispiel auf GitHub.

6.4 Verwendung des Beispiel-Consumers auf GitHub

Der unter https://github.com/brandmaker/MBI-Consumer bereitgestellte Beispiel-Consumer kann von interessierten Entwicklern verwendet werden. Er unterliegt keinerlei rechtlicher Beschränkung (Apache Commons License). BrandMaker übernimmt keine Gewährleistung, dass der veröffentlichte Quelltext fehlerfrei ist, noch dass dieser Anforderungen eines Verwenders erfüllen kann.

7 Dokumentation der verfügbaren Transaktionen je Modul

7.1 Job Manager

Hinweis

Die Nutzlast (Spalte: Provided payload data) kann variieren. Sie hängt davon ab, welche Felder im jeweiligen Job-Template konfiguriert sind.

Events im Job Manager werden anhand der Spalte „Category“ dedupliziert.

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 canceled	<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

Der Marketing Planer wird in einem der nächsten Releases von BrandMaker in den MBI integriert.

7.3 Review Manager

Der Review Manager wird in einem der nächsten Releases von BrandMaker in den MBI integriert.

7.4 Resource Management

Resource Management wird in einem der nächsten Releases von BrandMaker in den MBI integriert.

8 Sicherheitshinweise

8.1 SSL-Verschlüsselung

Der MBI verarbeitet ausschließlich URLs, die SSL-verschlüsseltes HTTP verwenden.

Der Betreiber des Consumers hat dafür Sorge zu tragen, dass ein gültiges SSL-Zertifikat auf seinem System installiert ist. Dieses muss von einer bekannten Root-CA signiert sein.

Selbstsignierte Zertifikate werden nicht akzeptiert und führen zu einem Verbindungsfehler und in Folge zu einer Deaktivierung des Consumers im MBI.

8.2 Anfrage-Validierung

Alle Anfragen gegen den Consumer werden vom MBI gemäß https://tools.ietf.org/id/draft-cavage-http-signatures-07.html signiert. Der Ersteller des Consumers ist zuständig, Anfragen anhand der Signatur und dem entsprechenden Public-Key des BrandMaker-Systems zu validieren, bevor sie verarbeitet werden und ungültige Anfragen zu verwerfen.

8.3 Anfragen an die BrandMaker-API

Ein MBI-Consumer ist nicht „per se“ autorisiert, Anfragen an die BrandMaker-APIs zu senden. Vielmehr ist dazu eine gültige Authentifizierung gegenüber dem BrandMaker-System notwendig. BrandMaker stellt dazu den OAuth2-konformen Dienst „BrandMaker CAS“ zur Verfügung. S.d.a. https://developers.brandmaker.com/guides/auth/

Bitte beachten sie, dass alle Anfragen an die APIs des BrandMaker-Systems immer im Kontext eines authentifizierten und autorisierten Benutzers erfolgen. Dessen Rechte und Rollen wirken auf das Ergebnis jeder Anfrage.