Entwicklerdokumentation

CC E-Commerce API – Version 1.1.3
etracker Campaign Control

Stand: September 2015
Version: 2.7

etracker®
more than analytics.

etracker hat bei der Erstellung der Dokumentation größtmögliche Sorgfalt walten lassen, trotzdem sind Fehler nicht ganz auszuschließen. Für die Mitteilung eventueller Fehler und Verbesserungsvorschläge sind wir jederzeit dankbar.

Es wird darauf hingewiesen, dass die in der Dokumentation verwendeten Soft- und Hardwarebezeichnungen sowie Markennamen der jeweiligen Firmen im Allgemeinen dem Warenzeichen-, Marken- oder patentrechtlichen Schutz unterliegen.

etracker GmbH
Erste Brunnenstraße 1
D-20459 Hamburg

Hotline: +49 40 55 56 59 77

Zentrale: +49 40 55 56 59 50
Fax: +49 40 55 56 59 59

Web: www.etracker.com
E-Mail: info@etracker.com

Inhaltsverzeichnis

1 Voraussetzungen und Ziele dieser Dokumentation 4

2 Funktionalität der CC E-Commerce API 5

3 Zugang zur CC E-Commerce API 5

4 Debug-Modus 6

4.1 Debug-Modus mit Konfigurationsobjekt _etr einschalten 6

4.2 Debug-Modus mit Variable etCommerce.debugMode einschalten 7

5 Die Funktionen der CC E-Commerce API 8

5.1 sendEvent - Event direkt absenden 8

5.2 attachEvent - Event an Objekt anknüpfen 9

5.3 Event zwischenspeichern (asynchroner Ablauf) 10

6 Mögliche Events 11

6.1 viewProduct - Produkt gesehen 11

6.2 insertToBasket - Produkt in den Warenkorb gelegt 13

6.3 removeFromBasket - Produkt aus dem Warenkorb entfernt 14

6.4 order - Bestellung 15

6.5 orderConfirmation - Bestellung vom Status Lead in den Status Sale transferiert 17

6.6 orderCancellation - Bestellung storniert 18

6.7 orderPartialCancellation - Bestellung teilstorniert 18

7 Event Objekte 20

7.1 Das Produkt-Objekt 20

7.2 Das Warenkorb-Objekt 22

7.3 Das Bestell-Objekt 23

8 Anwendungsbeispiele 26

8.1 Produktseite angesehen 26

8.2 Produkt in den Warenkorb gelegt 27

8.3 Ein Produkt aus dem Warenkorb entfernen (asynchroner Tracking Code) 28

8.4 Eine Bestellung beim Klick absenden (asynchroner Tracking Code) 29

9 Change log 30

9.1 Version 1.1.1 30

9.2 Version 1.1.2 30

9.3 Version 1.1.3 30

1 Voraussetzungen und Ziele dieser Dokumentation

Um das Shopsystem optimal an die Standard-Reports von Campaign Control (CC) anzupassen, können Sie mit Hilfe von sogenannten Events (Benutzeroberflächenereignissen) zusätzliche Kennzahlen erfassen, die über den Standard hinausgehen.

Diese Dokumentation beschreibt, wie verschiedene Kundenaktionen auf der Webseite (z. B. Bestellung, Kauf, Storno) als Events über die etracker CC E-Commerce API an etracker geschickt werden können.

Hinweis:
Den optionalen Funktionsparameter [Warenkorb-Id] können Sie zwar im Event-Code verwenden, er wird aber in den Reports noch nicht angezeigt. Ebenso können Sie folgende Attribute zwar schon in den Produkt- oder Bestell-Objekten definieren, jedoch noch nicht in den Reports sehen:

- currency
- variants
- pro_name
- ean
- customerId
- customerAddress
- invoiceAddress
- paymentMethod
- differenceData
- shipType
- shipCosts
- couponCodes
- giftPackage

In den Beispielen sind deshalb Parameter, die *in den Reports noch nicht zu sehen* sind, grau dargestellt.

Die Attribute customerGroup, deliveryConditions und paymentConditions sind nur im Product Performance Report sichtbar.

Diese Dokumentation richtet sich an Entwickler, die die Shop-Integration durchführen. Für das Verständnis werden JavaScript Kenntnisse, sowie die Kenntnis der benötigten Kennzahlen vorausgesetzt, damit die dafür notwendigen Events in den Code integriert werden können.

Die CC E-Commerce API liegt derzeit in der **Version 1.1.3** vor. Nach Veröffentlichung einer neuen API Version wird diese API Version nur noch 6 Monate unterstützt. etracker wird Sie rechtzeitig informieren, wenn die API Version sich ändert.

2 Funktionalität der CC E-Commerce API

Die CC E-Commerce API ist eine JavaScript-Schnittstelle. Sie kann verschiedene Events, die ein Kunde in einem Online-Shop auslöst, entgegennehmen und an etracker übermitteln. Dies kann klassisch pro Seitenaufruf oder auch Event-gesteuert geschehen. Die dabei übertragenen Daten bestehen aus Informationen über die Produkte, Warenkörbe und Bestellungen. Mit den so erfassten Daten können Sie z. B. ermitteln, wie viele Warenkörbe stehengelassen wurden oder welche Produkte besonders häufig angeschaut wurden.

Die Datenübermittlung kann zum tatsächlichen Zeitpunkt der Kundenaktion geschehen, wenn beispielsweise der Kunde das Produkt in den Warenkorb legt, oder später, z. B. im Fall eines abgesendeten Formulars. Hier wird das Event erst gesendet, wenn der Kunde auf der Folgeseite angelangt ist. In einer asynchronen Umgebung ist es außerdem möglich, Events zu speichern und erst an etracker zu senden, wenn der Tracking Code geladen ist. Die Beispiele weiter unten erläutern die verschiedenen Varianten zum Einbau von Events.

Hinweis:
Wenn Sie Bestellungen über Web Analytics *und* die E-Commerce API an etracker übergeben, werden die Daten aus Web Analytics ab einem bestimmten Prozentsatz automatisch in Campaign Control unterdrückt, damit die Bestellungen in den Campaign Control Reports nicht doppelt gezählt werden.

3 Zugang zur CC E-Commerce API

Hinweis:
Wir empfehlen, für die Entwicklung und den Einbau der API einen Test-Account bei etracker zu beantragen.

Bevor Sie Campaign Control, die CC E-Commerce API und alle anderen etracker Produkte einsetzen können, müssen Sie den aktuellen etracker Tracking Code, welchen Sie innerhalb Ihres etracker Accounts unter **Einstellungen > Setup/Tracking-Code** finden, auf allen zu messenden Seiten der Website einbinden. Zudem empfehlen wir Ihnen, den Parameter et_pagename zu setzen, damit die CC Reports die einzelnen Seiten Ihrer Website anzeigen können. Weitere Konfigurationsmöglichkeiten des etracker Tracking Codes werden im Technischen Handbuch Kapitel 3.3 ausführlich erläutert.

Die Schnittstelle ist mit der Auslieferung des etracker Tracking Codes aktiviert, sodass Sie die Events sofort in den Code einbinden können.

4 Debug-Modus

Die Schnittstelle ist mit einem Debug-Modus ausgestattet, der besonders während des Einbaus der CC E-Commerce API hilfreich ist. Wenn der Debug-Modus aktiviert ist, werden Fehlermeldungen, Events und der Aufruf der etracker-Schnittstelle in der JavaScript-Konsole angezeigt. Die Events werden *nicht* an etracker übermittelt, wenn der Debug-Modus aktiviert ist.

Der Modus kann erst eingeschaltet werden, wenn der Tracking Code komplett geladen ist.

Hinweis:
Alle zu definierenden Elemente sind in den Code-Beispielen grün markiert.

Um die Ausgaben lesen zu können, helfen Entwicklertools, wie sie heute von vielen Browsern angeboten werden.

Browser	Entwicklertool	Bezugsort
Firefox	Firebug	https://addons.mozilla.org/de/firefox/addon/firebug
Internet Explorer	Entwicklertools	Ist im Browser integriert
Chrome	Entwicklertools	Ist im Browser integriert
Opera	Dragonfly	Ist im Browser integriert
Safari	Firebug lite	http://getfirebug.com/firebuglite

4.1 Debug-Modus mit Konfigurationsobjekt _etr einschalten

Hinweis:
Das Konfigurationsobjekt können Sie nur einbauen, wenn Sie den aktuellen Tracking Code 4.x verwenden. Die Einstellung greift erst, wenn die Datei e.js nachgeladen ist. Vorher versendete Events werden aber trotzdem auf Fehler geprüft, da sie ja erst verarbeitet werden, wenn die e.js geladen ist.
_etr sollte nicht überschrieben werden, falls es bereits existiert.

Beispiel:

```
<script type="text/javascript" charset="UTF-8">
    // Parameterblock
    var et_pagename      = "__INDEX__";
    _etr = {
      debugMode = true;
    }
</script>
```

```
<script id="_etLoader" type="text/javascript" charset="UTF-8" data-
preload-dc="data-preload-dc" data-secure-code="Account Schlüssel 1"
src="//static.etracker.com/code/e.js"></script>
```

4.2 Debug-Modus mit Variable etCommerce.debugMode einschalten

Hinweis:
Die Variable etCommerce.debugMode muss *nach* dem Aufruf des Tracking Codes gesetzt werden. Der Tracking Code 4.x wird durch die Datei e.js aufgerufen, der Tracking Code 3.x durch die Datei t.js.

```
// einschalten des Debug-Modus
etCommerce.debugMode = true ;
```

Beispiel für Tracking Code 4.x
etCommerce.debugMode *hinter* Parameterblock und Tracklet:

```
<script type="text/javascript" charset="UTF-8">
    // Parameterblock
    var et_pagename     = "Seitenname";
</script>
<script id="_etLoader" type="text/javascript" charset="UTF-8" data-
preload-dc="data-preload-dc" data-secure-code="Account Schlüssel 1"
src="//static.etracker.com/code/e.js"></script>
<script type="text/javascript" charset="UTF-8">
    var etCommerce.debugMode = true ;
</script>
```

Beispiel für Tracking Code 3.x
etCommerce.debugMode *im* Parameterblock:

```
<script type="text/javascript"
src="https://code.etracker.com/t.js?et=#Account Schlüssel 1#"></script>
<script type="text/javascript" charset="UTF-8">
    // Parameterblock
    var et_pagename     = "Seitenname";
    var etCommerce.debugMode = true ;
    _etc();
</script>
```

5 Die Funktionen der CC E-Commerce API

Die Schnittstelle hat zwei grundlegende Funktionen, um Informationen an etracker zu übermitteln: sendEvent und attachEvent. sendEvent ist der direkte Aufruf eines von der Schnittstelle definierten E-Commerce Events, welcher sofort die übergebenen Werte sendet. Soll das Absenden an ein bestimmtes JavaScript-Event gekoppelt sein – z. B. Besucher klickt auf „in den Warenkorb“ – dann kann die Funktion attachEvent verwendet werden, die das E-Commerce Event an ein gewünschtes Objekt der Webseite anhängt.

Wenn der Tracking Code am Ende der HTML-Seite eingebaut ist oder asynchron geladen wird, gibt es die Möglichkeit, Events und das Anhängen von Events an HTML-Objekte zwischen zu speichern. Die Funktionen werden dann ausgeführt, wenn der etracker Tracking Code komplett geladen ist.

5.1 sendEvent - Event direkt absenden

```
etCommerce.sendEvent(event, parameter_1, [parameter_n])
```

Die Funktion sendEvent des Objekts etCommerce wird im JavaScript direkt aufgerufen. Die übergebenen Werte werden dabei direkt an etracker übermittelt.

Funktionsparameter	Datentyp	Begrenzung	Beschreibung
event	string	Es werden nur die von der Schnittstelle definierten Events unterstützt. Siehe Kapitel 6	Name des Events
parameter_1, [parameter_n]	variiert	Siehe weitere Beschreibung	

Hinweis: Parameter in eckigen Klammern [ ] sind optionale Parameter.

Beispiel:

```
// direkter Aufruf
etCommerce.sendEvent('viewProduct', product, 'Warenkorb 1') ;
```

5.2 attachEvent - Event an Objekt anknüpfen

```
etCommerce.attachEvent(attachObject, event, parameter_1, [parameter_n])
```

Mit der Funktion attachEvent können Sie an jedes Webseiten-Objekt, welches eine ID besitzt, ein E-Commerce Event anhängen. Dieses wird dann durch das angegebene JavaScript-Event ausgelöst. So kann direkt mit dem Klick auf den Button „In den Warenkorb“ das E-Commerce Event an etracker übermittelt werden.

Funktionsparameter	Datentyp	Begrenzung	Beschreibung
attachObject	Objekt	Es werden nur bestehende JavaScript-Events und Objekt-IDs, die mittels getElementById ermittelt werden, unterstützt. Der Aufbau des Objekts ist folgendermaßen: {'Eventname' : ['Objekt-ID1', ‘Objekt-ID2’]}	In diesem Objekt sind das JavaScript-Event und die IDs der Webseiten-Objekte enthalten, an das dieses Event angehängt wird.
event	String	Es werden nur die von der Schnittstelle definierten Events unterstützt. Siehe Kapitel 6	Name des angehängten Events
parameter_1, [parameter_n]	variiert	Siehe weitere Beschreibung	

Hinweis: Parameter in eckigen Klammern [ ] sind optionale Parameter.

Beispiel:

```
// Verknüpfen des E-Commerce Events mit einem JavaScript-Event
etCommerce.attachEvent({'mousedown' : ['viewButton']}, 'viewProduct', product, 'Warenkorb 1') ;
```

5.3 Event zwischenspeichern (asynchroner Ablauf)

```
var etCommercePrepareEvents = [] ;
etCommercePrepareEvents.push(eventArray);
```

Wenn Events abgesendet oder an Objekte gehängt werden sollen, bevor der etracker Tracking Code geladen ist, können diese Aufgaben zwischengespeichert werden. Sie werden dann erst ausgeführt, wenn der Tracking Code und die Seite komplett geladen sind. Zu diesem Zweck müssen Sie einmalig ein JavaScript-Array namens etCommercePrepareEvents definieren, welches anschließend von der CC E-Commerce API verwendet wird.

Funktionsparameter	Datentyp	Begrenzung	Beschreibung
eventArray	array	Siehe folgende Tabelle (Aufbau des Event-Arrays)	In diesem Array sind die Funktionsnamen des gewünschten Events und die passenden Parameter enthalten

Das Event-Array ist folgendermaßen aufgebaut:

Name	Datentyp	Begrenzung	Kommentar
Event-Funktion	string	sendEvent oder attachEvent	Die jeweilige Funktion, welche aufgerufen werden soll
Weitere Parameter wie der jeweilige Funktionsaufruf	variiert	Wie in Kapitel 5.2 beschrieben	Die Parameter werden als Array-Elemente dem Funktionsnamen hinten angestellt.

Die Aufrufparameter der Funktionen sendEvent und attachEvent ändern sich nicht, sie werden in diesem Fall lediglich durch eckige Klammern eingefasst. Zusätzlich muss die gewünschte Funktion angegeben werden.

Beispiele:

```
// Definition des JavaScript Arrays für den asynchronen Ablauf
var etCommercePrepareEvents = [] ;

// Zwischenspeichern des Events im asynchronen Ablauf
etCommercePrepareEvents.push(['sendEvent', 'viewProduct', product, 'Warenkorb 1']);

// Zwischenspeichern der Verknüpfung des Aufrufes mit einem Event
etCommercePrepareEvents.push(['attachEvent', {'mousedown' : ['viewButton']}, 'viewProduct',
product, 'Warenkorb 1') ;
```

6 Mögliche Events

Gegenwärtig unterstützt die CC E-Commerce API sechs verschiedene Events.

6.1 viewProduct - Produkt gesehen

Dieses Event kann gesendet werden, wenn sich der Kunde auf der Produktseite oder einer Übersichtsseite von Produkten befindet.

Funktionsparameter	Datentyp	Begrenzung	Beschreibung
'viewProduct'	string	Nur dieser Name ist zugelassen	Name des Events
Produkt-Objekt	object	Das Objekt muss der Produkt-Objekt-Beschreibung entsprechen siehe Kapitel 7.1	Ein Produkt-Objekt
[Warenkorb-Id]	string	maximal 50 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	Falls schon eine Warenkorb-ID existiert, kann diese übergeben werden um stehen gelassene Warenkörbe zu messen
[Seitenname]	string	maximal 255 Zeichen lang Leerzeichen am Anfang und Ende werden entfernt	Unterscheidet sich der Seitenname von dem Standard, kann dieser hier übergeben werden

Hinweis:
Parameter in eckigen Klammern [ ] sind optionale Parameter. Die Warenkorb-Id wird noch nicht in Reports angezeigt.

Beispiele:

```
// Definition des Produkt-Objekts
var product =
{
        id          : '3445',
        name        : 'Elfrida',
        category : ['Tiere', 'Großwild', 'Giraffen', 'Liebe Giraffen'],
        price       : '1723.60',
        currency : 'EUR',
        variants : {'Farbe' : 'gelb', 'Geschlecht' : 'weiblich', 'Figur' : 'dünn'}
};
```

```
// direkter Aufruf
etCommerce.sendEvent('viewProduct', product, 'Warenkorb 1') ;
```

```
// Als Event angehängt
etCommerce.attachEvent({'mousedown' : ['button_ view']}, 'viewProduct', product, 'Warenkorb 1') ;
```

```
// asynchroner Aufruf
etCommercePrepareEvents.push(['sendEvent', 'viewProduct', product, 'Warenkorb 1']) ;
```

```
// asynchron angehängtes Event
etCommercePrepareEvents.push(['attachEvent', {'mousedown' : ['button_ view']}, 'viewProduct',
product, 'Warenkorb 1']) ;
```

6.2 insertToBasket - Produkt in den Warenkorb gelegt

Dieses Event wird gesendet, wenn der Kunde seinem Warenkorb ein Produkt hinzufügt.

Funktionsparameter	Datentyp	Begrenzung	Beschreibung
'insertToBasket'	string	Nur dieser Name ist zugelassen	Name des Events
Produkt-Objekt	object	Das Objekt muss der Produkt-Objekt-Beschreibung entsprechen Siehe Kapitel 7.1	Ein Produkt-Objekt
Anzahl	integer	0 - 65 535 Negative Zahlen sind nicht erlaubt	Die Anzahl der in den Warenkorb gelegten Produkte
[Warenkorb-Id]	string	maximal 50 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	Falls schon eine Warenkorb-ID existiert, kann diese übergeben werden, um stehen gelassene Warenkörbe zu zählen
[Seitenname]	string	maximal 255 Zeichen lang Leerzeichen am Anfang und Ende werden entfernt	Unterscheidet sich der Seitenname von dem Standard, kann dieser Seitenname übergeben werden

Hinweis:
Parameter in eckigen Klammern [ ] sind optionale Parameter. Die Warenkorb-Id wird noch nicht in Reports angezeigt.

Beispiele:

```
// direkter Aufruf
etCommerce.sendEvent('insertToBasket', product, 2, 'Warenkorb 1') ;
```

```
// Als Event angehängt
etCommerce.attachEvent({'mousedown' : ['insertButton']}, 'insertToBasket', product, 2, 'Warenkorb 1') ;
```

```
// asynchroner Aufruf
etCommercePrepareEvents.push(['sendEvent', 'insertToBasket', product, 2, 'Warenkorb 1']) ;
```

```
// asynchron angehängtes Event
etCommercePrepareEvents.push(['attachEvent', {'mousedown' : ['insertButton']}, 'insertToBasket',
product, 2, 'Warenkorb 1']) ;
```

6.3 removeFromBasket - Produkt aus dem Warenkorb entfernt

Mit Hilfe dieses Events können Sie feststellen, welche Produkte wieder aus einem Warenkorb entfernt wurden.

Funktionsparameter	Datentyp	Begrenzung	Beschreibung
'removeFromBasket'	string	Nur dieser Name ist zugelassen	Name des Events
Produkt-Objekt	object	Das Objekt muss der Produkt-Objekt-Beschreibung entsprechen (s. Kapitel 7.1)	Ein Produkt-Objekt
Anzahl	integer	0 - 65 535 Negative Zahlen sind nicht erlaubt	Die Anzahl der entfernten Produkte
[Warenkorb-Id]	string	maximal 50 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	Falls schon eine Warenkorb-ID existiert, kann diese übergeben werden, um stehen gelassene Warenkörbe zu messen
[Seitenname]	string	maximal 255 Zeichen lang Leerzeichen am Anfang und Ende werden entfernt	Unterscheidet sich der Seitenname von dem Standard, kann dieser Seitenname übergeben werden

Hinweis:
Parameter in eckigen Klammern [ ] sind optionale Parameter. Die Warenkorb-Id wird noch nicht in Reports angezeigt.

Beispiele:

```
// direkter Aufruf
etCommerce.sendEvent('removeFromBasket', product, 1, 'Warenkorb 1') ;
```

```
// Als Event angehängt
etCommerce.attachEvent({'mousedown' : ['removeButton']}, 'removeFromBasket', product, 1,
'Warenkorb 1') ;
```

```
// asynchroner Aufruf
etCommercePrepareEvents.push(['sendEvent', 'removeFromBasket', product, 1, 'Warenkorb 1']) ;
```

```
// asynchron angehängtes Event
etCommercePrepareEvents.push(['attachEvent', {'mousedown' : ['removeButton']},
'removeFromBasket', product, 1, 'Warenkorb 1']) ;
```

6.4 order - Bestellung

Dieses Event übermittelt die gesamte Bestellung mit allen Bestelldaten und dem Warenkorb.

Funktionsparameter	Datentyp	Begrenzung	Beschreibung
'order'	string	Nur dieser Name ist erlaubt	Name des Events
Bestell-Objekt	object	Das Objekt muss der Bestell-Objekt-Beschreibung entsprechen. Siehe Kapitel 7.3	Ein Bestell-Objekt
[Seitenname]	string	maximal 255 Zeichen lang Leerzeichen am Anfang und Ende werden entfernt	Unterscheidet sich der Seitenname vom Standard, kann dieser Seitenname übergeben werden

Hinweis: Parameter in eckigen Klammern [ ] sind optionale Parameter.

Beispiele:

```
// Definition des Bestell-Objekts
var order =
{
        orderNumber : 'Bestellnummer 1',
        status : 'sale',
        orderPrice : '1723.60',
        currency : 'EUR',
        basket : {
                id : 'Warenkorb 1',
                products : [
                {
                        product: {
                                id          : '3445',
                                name        : 'Elfrida',
                                category : ['Tiere', 'Großwild', 'Giraffen', 'Liebe Giraffen'],
                                price       : '1723.60',
                                currency : 'EUR',
                                variants : {'Farbe' : 'gelb', 'Geschlecht' : 'weiblich', 'Figur' : 'dünn'}
                        },
                        quantity : 1
                }]
        },
        customerId : '345465',
        customerGroup : 'Tierliebhaber',
        customerAddress : 'Deutschland,Hamburg,Hamburg',
        invoiceAddress : 'Deutschland,Baden-Württemberg,Stuttgart',
        paymentMethod : 'Paypal',
        deliveryConditions : 'Großer Transport',
        paymentConditions : 'Sofortzahlung',
        shipType : 'Spedition',
        shipCosts : '1672',
        couponCodes : '43847884595',
        giftPackage : ['3445'],
        differenceData : 0
};
```

```
// direkter Aufruf
etCommerce.sendEvent('order', order) ;
```

```
// Als Event angehängt
etCommerce.attachEvent({'mousedown' : ['orderButton']}, 'order', order) ;
```

```
// asynchroner Aufruf
etCommercePrepareEvents.push(['sendEvent', 'order', order]) ;
```

```
// asynchron angehängtes Event
etCommercePrepareEvents.push(['attachEvent', {'mousedown' : ['orderButton']}, 'order', order]) ;
```

6.5 orderConfirmation - Bestellung vom Status Lead in den Status Sale transferiert

Dieses Event wird gesendet, wenn eine Bestellung inklusive aller Produktpositionen vom Status "Lead" in den Status "Sale" überführt werden soll.

Funktionsparameter	Datentyp	Begrenzung	Beschreibung
'orderConfirmation'	string	Nur dieser Name ist zugelassen	Name des Events
Bestellnummer	string	maximal 50 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	Die Bestellnummer der Bestellung, für die der Status "Lead" in "Sale" übergehen soll

Beispiele:

```
// direkter Aufruf
etCommerce.sendEvent('orderConfirmation', 'Bestellnummer 1');
```

```
// Als Event angehängt
etCommerce.attachEvent({'mousedown' :['cancelButton']}, 'orderConfirmation', 'Bestellnummer 1') ;
```

```
// asynchroner Aufruf
etCommercePrepareEvents.push(['sendEvent','orderConfirmation', 'Bestellnummer 1']) ;
```

```
// asynchron angehängtes Event
etCommercePrepareEvents.push(['attachEvent', {'mousedown' : ['cancelButton']},
'orderConfirmation', 'Bestellnummer 1']) ;
```

6.6 orderCancellation - Bestellung storniert

Dieses Event wird gesendet, wenn der Kunde die gesamte Bestellung storniert.

Funktionsparameter	Datentyp	Begrenzung	Beschreibung
'orderCancellation'	string	Nur dieser Name ist zugelassen	Name des Events
Bestellnummer	string	maximal 50 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	Die Bestellnummer der Bestellung, die storniert werden soll

Beispiele:

```
// direkter Aufruf
etCommerce.sendEvent('orderCancellation', 'Bestellnummer 1') ;
```

```
// Als Event angehängt
etCommerce.attachEvent({'mousedown' : ['cancelButton']}, 'orderCancellation', 'Bestellnummer 1') ;
```

```
// asynchroner Aufruf
etCommercePrepareEvents.push(['sendEvent', 'orderCancellation', 'Bestellnummer 1']) ;
```

```
// asynchron angehängtes Event
etCommercePrepareEvents.push(['attachEvent', {'mousedown' : ['cancelButton']},
'orderCancellation', 'Bestellnummer 1']) ;
```

6.7 orderPartialCancellation - Bestellung teilstorniert

Dieses Event wird gesendet, wenn nur einzelne Produkte einer Bestellung storniert werden.

Funktionsparameter	Datentyp	Begrenzung	Beschreibung
'orderPartialCancellation'	string	Nur dieser Name ist erlaubt.	Name des Events
Bestellnummer	string	maximal 50 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	Die Bestellnummer der Bestellung, die storniert werden soll
Produkt-Objekte	array of objects	Das Array muss der Produkte-Array-Beschreibung entsprechen Siehe Kapitel 7.2	Ein Array, welches aus verschiedenen Produkt-Objekten und mit ihrer jeweils zu stornierenden Anzahl besteht

Beispiele:

```
// Definition von Produkt-Objekten
var products = [
	{
		product : {
			id		: '3445',
			name	: 'Elfrida',
			category : ['Tiere', 'Großwild', 'Giraffen', 'Liebe Giraffen'],
			price	: '1723.60',
			currency : 'EUR',
			variants : {'Farbe' : 'gelb', 'Geschlecht' : 'weiblich', 'Figur' : 'dünn'}
		},
		quantity: 1
	}
];
```

```
// direkter Aufruf
etCommerce.sendEvent('orderPartialCancellation', 'Bestellnummer 1' , products) ;
```

```
// Als Event angehängt
etCommerce.attachEvent({'mousedown' : ['partialButton']}, 'orderPartialCancellation',
'Bestellnummer 1' , products) ;
```

```
// asynchroner Aufruf
etCommercePrepareEvents.push(['sendEvent', 'orderPartialCancellation', 'Bestellnummer 1' ,
products]) ;
```

```
// asynchron angehängtes Event
etCommercePrepareEvents.push(['attachEvent', {'mousedown' : ['partialButton']},
'orderPartialCancellation', 'Bestellnummer 1' , products]) ;
```

7 Event Objekte

Die Informationen zu den Produkten, den Warenkörben und den Bestellungen werden in JavaScript-Objekten abgelegt. Im Folgenden finden Sie eine Aufstellung dieser Objekte und ihren Aufbau.

7.1 Das Produkt-Objekt

Dieses Objekt definiert ein Produkt mit den dazugehörigen Attributen.

Name	Attribut	Datentyp	Begrenzung	Kommentar
Produkt-ID	id	string	maximal 50 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	Die Produkt-ID wird von Ihnen festgelegt und ergibt sich z. B. aus Ihrem Warenwirtschaftssystem.
Produkt-Name	name	string	maximal 255 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	Der Name des Produktes.
Produkt-Hierarchie (Kategorie)	category	array of strings	Es können maximal vierstufige Hierarchien abgebildet werden. Das Array oder eine Kategorie kann auch leer sein. Die Hierarchien können 50 Zeichen lang sein, Leerzeichen am Anfang und Ende werden entfernt.	Die Produkthierachie wird in einem Array gespeichert. z. B.: ['Monitore', '', 'Flachbildschirme', 'LED']
(Nominal)-Preis	price	string	maximal 20 Zeichen lang, Dezimaltrenner ist ein Punkt. Leerzeichen am Anfang und Ende werden entfernt	Der Preis des Produktes
Währung	currency	string	maximal 3 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	Die Währung nach ISO 4217 z. B.: EUR oder USD
Varianten	variants	Object with key/value pairs	Das Objekt kann leer sein. Die Varianten können 50 Zeichen lang sein, Leerzeichen am Anfang und Ende werden entfernt. Maximal 3 Varianten	Um verschiedene Varianten eines Produktes zu übergeben. z. B.: {'Farbe' : 'gelb', 'Geschlecht' : 'weiblich', 'Figur' : 'dünn'}
[Lieferanten/ Hersteller-name]	pro_name	string	maximal 100 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	Name des Lieferanten oder Herstellers
[EAN]	ean	string	maximal 20 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	European Article Number

Hinweis:
Parameter in eckigen Klammern [ ] sind optionale Parameter. Die Attribute currency, variants, pro_name und ean sind noch nicht in den Reports sichtbar.

Beispiel:

```
// Definition eines Produkt-Objekts
var product =
{
        id          : '3445',
        name        : 'Elfrida',
        category : ['Tiere', 'Großwild', 'Giraffen', 'Liebe Giraffen'],
        price       : '1723.60',
        currency : 'EUR',
        variants : {'Farbe' : 'gelb', 'Geschlecht' : 'weiblich', 'Figur' : 'dünn'}
};
```

7.2 Das Warenkorb-Objekt

Bei einer Bestellung werden die bestellten Produkte in einem Warenkorb-Objekt abgelegt.

Name	Attribut	Datentyp	Begrenzung	Kommentar
Warenkorb-ID	id	string	maximal 50 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	Die Warenkorb-ID wird von Ihnen festgelegt.
Produkt-Objekte	products	array of objects	Das Array muss der Produkte-Array-Beschreibung entsprechen Siehe folgende Tabelle	In diesem Array werden die verschiedenen Produkt-Objekte und die bestellte Anzahl hinterlegt.

Das Array *products* enthält ein oder mehrere Objekte die wiederum aus Produkt-Objekten und der jeweiligen Anzahl bestehen. Dieses Produkte-Array findet auch in einer Teilstornierung Verwendung.

Name	Attribut	Datentyp	Begrenzung	Kommentar
Produkt	product	object	Das Objekt muss der Produkt-Objekt-Beschreibung entsprechen	Das Produkt-Objekt
Anzahl	quantity	integer	0 - 65 535 Negative Zahlen sind nicht erlaubt	Die bestellte/stornierte Anzahl

Beispiel:

```
// Definition eines Warenkorb-Objekts mit Produkte-Array
var basket =
{
  id : '3',
  products : [
    {
            product: {
            …
            },
            quantity : 2
    },
    {
            product: {
            …
            },
            quantity : 1
    }
  ]
};
```

7.3 Das Bestell-Objekt

Das Objekt der Bestellung enthält sämtliche Bestelldaten und das Warenkorb-Objekt

Name	Attribut	Datentyp	Begrenzung	Kommentar
Bestell-nummer	orderNumber	string	maximal 50 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	Die von Ihnen festgelegte **eindeutige** Bestellnummer. Mit dieser werden spätere Stornierungen getätigt. Bestellnummern, die sich nicht eindeutig einer Bestellung zuordnen lassen, verfälschen die Daten.
Status	status	enum	Enthält lead, sale, cancellation oder partial_cancellation	
Bestellwert	orderPrice	string	maximal 20 Zeichen lang, Dezimaltrenner ist ein Punkt. Leerzeichen am Anfang und Ende werden entfernt	Der Gesamtwert der Bestellung. Er sollte sich möglichst als Summe aus Warenkorbwert[1] und Versandkosten ergeben. Rabatte, Gutscheine oder Sonderkosten durch Zahlungsart o.ä. sollten als Produkt-Objekt erfasst werden.
Währung	currency	string	maximal 3 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	Die Währung der Bestellung nach ISO 4217 z. B.: EUR oder USD
Warenkorb	basket	object of warenkorb	Das Objekt muss der Warenkorb-Objekt-Beschreibung entsprechen	Das Warenkorb-Objekt
[Kunden-Id]	customerId	string	maximal 50 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	Eine eindeutige ID dieses Kunden

[1] Der Warenkorbwert ist die Summe der Werte für die einzelnen Produkte abzüglich Rabatte, Gutscheine etc.

[Kunden-gruppe]	customerGroup	string	maximal 50 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	z. B. ■ Neukunde ■ Stammkunde ■ Vielkäufer ■ VIP
[Lieferadresse/ Geolokation]	customerAddress	string	maximal 255 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	
[Rechnungs-adresse/ Geolokation]	invoiceAddress	string	maximal 255 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	
[Bezahlungs-art]	paymentMethod	string	maximal 50 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	z. B. ■ Vorabüberweisung ■ PayPal ■ Sofortüberweisung ■ Giropay ■ Kreditkarte ■ Rechnung ■ Ratenzahlung
[Liefer-bedingungen]	deliveryConditions	string	maximal 255 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	z. B. ■ Lieferung bis Bordsteinkante ■ Aufstellung vor Ort ■ Lieferung an Packstation/Paket-shop/Filiale
[Zahlungs-bedingungen]	paymentConditions	string	maximal 255 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	z. B. ■ Spezielle Zahlungsziele ■ Skonto ■ Ratenzahlung
[Versandart]	shipType	string	maximal 50 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	In der Regel Kombination aus Versender und Zeitraum.
[Versand-kosten]	shipCosts	string	maximal 20 Zeichen lang, Dezimaltrenner ist ein Punkt. Leerzeichen am Anfang und Ende werden entfernt	

[Gutschein-codes]	couponCodes	string	maximal 50 Zeichen lang, Leerzeichen am Anfang und Ende werden entfernt	
[Geschenk-verpackung]	giftPackage	array of product id's	Das Array kann leer sein. Produkt-Ids können maximal 50 Zeichen lang sein, Leerzeichen am Anfang und Ende werden entfernt	z. B. ['1384', '304']
[Differenz-daten]	differenceData	boolean	true oder false	Hiermit kann definiert werden, ob die Bestellung komplett ist oder nur einen Teil des Warenkorbs beinhaltet.

Hinweis:
Parameter in eckigen Klammern [ ] sind optionale Parameter. Folgende Attribute sind noch nicht in den Reports sichtbar:
currency, customerId, customerAddress, invoiceAddress, paymentMethod, differenceData, shipType, shipCosts, couponCodes und giftPackage.
Die Attribute customerGroup, deliveryConditions und paymentConditions sind nur im Product Performance Report sichtbar.

Beispiel:

```
// Definition eines Bestell-Objekts
var order =
{
        orderNumber : '234',
        status : 'sale',
        orderPrice : '5447.2',
        currency : 'EUR',
        basket : {Warenkorb-Objekt},
        customerId : '345465',
        customerGroup : ‘Tierliebhaber’,
        customerAddress : ‘Deutschland,Hamburg,Hamburg’,
        invoiceAddress : ‘Deutschland,Baden-Württemberg,Stuttgart’,
        paymentMethod : 'Paypal',
        deliveryConditions : 'Großer Transport',
        paymentConditions : 'Sofortzahlung',
        shipType : 'Spedition',
        shipCosts : '1672',
        couponCodes : '43847884595',
        giftPackage : ['3445'],
        differenceData : 0
}
```

8 Anwendungsbeispiele

Die Anwendungsbeispiele verdeutlichen, wie die verschiedenen Events für ausgewählte Aktionen auf der Webseite an etracker gesendet werden können.

8.1 Produktseite angesehen

Beim Aufrufen einer Produktseite sollen die Produktinformationen sofort an etracker übermittelt werden. Dafür wird ein Produkt-Objekt definiert. Die Daten werden über die „sendEvent“ Funktion direkt übermittelt.

Beispiel:

```
var et_Commerce_basketId = '3';

var et_Commerce_product =
{
        id          : '3445',
        name        : 'TV 47 Zoll Special Angebot',
        category : [‘TV’, 'LCD', '47', 'Special'],
        price       : '1723.60',
        currency : 'EUR',
        variants : {'Größe' : '47', 'Display' : 'LCD', 'Extras' : 'Standfuß'}
};

// Produkt gesehen
etCommerce.sendEvent('viewProduct', et_Commerce_product, et_Commerce_basketId);
```

8.2 Produkt in den Warenkorb gelegt

Um Produkte zu erfassen, die in den Warenkorb gelegt werden (durch Klicken auf einen „In den Warenkorb legen“-Button), ist ein Event zu definieren, das sich an den vorhandenen Button hängt. Zuvor muss ein Produkt-Objekt definiert werden, das die Produktdaten enthält. Die ID des Buttons lautet in diesem Fall „ButtonAddToBasket“ und die Datenübermittlung erfolgt, wenn das JavaScript-Event „mousedown“ ausgelöst wird. Die Anzahl der erfassten Produkte ergibt sich aus einem auf der Webseite hinterlegten Formularfeld „ProductQuantity“.

Beispiel:

```
var et_Commerce_basketId = '3';

var et_Commerce_product =
{
        id          : '3445',
        name        : 'TV 47 Zoll Special Angebot',
        category : [‘TV’, 'LCD', '47', 'Special'],
        price       : '1723.60',
        currency : 'EUR',
        variants : {'Größe' : '47', 'Display' : 'LCD', 'Extras' : 'Standfuß'}
};

Var et_Commerce_quantity = Number(document.getElementById('ProductQuantity').value) ;

etCommerce.attachEvent({'mousedown' : ['ButtonAddToBasket']}, 'insertToBasket',
et_Commerce_product, et_Commerce_quantity, et_Commerce_basketId);
```

8.3 Ein Produkt aus dem Warenkorb entfernen (asynchroner Tracking Code)

Um die Information zu übermitteln, dass der Kunde ein Produkt wieder aus dem Warenkorb genommen hat, muss das Event „removeFromBasket“ aufgerufen werden. Auf einer Seite, die den asynchronen Tracking Code verwendet, sind die etCommerce-Funktionen zum Zeitpunkt des Aufrufes im Quelltest noch nicht verfügbar, da diese erst mit der komplett geladenen Webseite bereit stehen. Zu diesem Zweck müssen die Events zwischengespeichert werden. Dafür steht ein JavaScript-Array zur Verfügung, welches in der CC E-Commerce API ausgewertet wird, sobald die Seite vollständig geladen ist.

Beispiel:

```
var et_Commerce_basketId = '3';

var et_Commerce_product =
{
        id          : '3445',
        name        : 'TV 47 Zoll Special Angebot',
        category : [‘TV’, 'LCD', '47', 'Special'],
        price       : '1723.60',
        currency : 'EUR',
        variants : {'Größe' : '47', 'Display' : 'LCD', 'Extras' : 'Standfuß'}
};

var etCommercePrepareEvents = [] ;

etCommercePrepareEvents.push(['sendEvent', 'removeFromBasket', et_Commerce_product, 1,
et_Commerce_basketId]);
```

8.4 Eine Bestellung beim Klick absenden (asynchroner Tracking Code)

Um eine Bestellung direkt beim Klicken auf den Bestellknopf an etracker zu melden, benötigt man ein Event „order“, das an den Button „sendOrder“ angehängt wird. Da der Tracking Code zuletzt geladen wird, muss das Anhängen im Array etCommercePrepareEvents zwischengespeichert werden.

Beispiel:

```
var basket =
{
        id : '3',
        products : [
        {
                product:
                {
                        id          : '3445',
                        name        : 'Elfrida',
                        category : ['Tiere', 'Großwild', 'Giraffen', 'Liebe Giraffen'],
                        price       : '1723.60',
                        currency : 'EUR',
                        variants : {'Farbe' : 'gelb', 'Geschlecht' : 'weiblich', 'Figur' : 'dünn'}
                },
                quantity : 1
        }]
}

var order =
{
        orderNumber : '234',
        status : 'sale',
        orderPrice : '5447.2',
        currency : 'EUR',
        basket : basket,
        customerId : '345465',
        customerGroup : ‘Tierliebhaber’,
        customerAddress : ‘Deutschland,Hamburg,Hamburg’,
        invoiceAddress : ‘Deutschland,Baden-Württemberg,Stuttgart’,
        paymentMethod : 'Paypal',
        deliveryConditions : 'Großer Transport',
        paymentConditions : 'Sofortzahlung',
        shipType : 'Spedition',
        shipCosts : '1672',
        couponCodes : '43847884595',
        giftPackage : ['3445'],
        differenceData : 0
}

var etCommercePrepareEvents = [] ;

etCommercePrepareEvents.push(['attachEvent', {'mousedown' : ['sendOrder']}, 'order', order]);
```

9 Change log

Hier sind die Änderungen an der Schnittstelle aufgelistet. Die Anwendungsbeispiele verdeutlichen, wie die verschiedenen Events für ausgewählte Aktionen auf der Webseite an etracker gesendet werden können.

9.1 Version 1.1.1

- Fehler im Debug-Modus behoben
- Die Definition von Objekten, die bei einem gewissen Browser-Event ausgelöst werden, hat sich geändert (attachEvent Funktion). Ab jetzt können Sie pro Browser-Event verschiedene Objekt-IDs in einem Array definieren.

```
// Alte Darstellungsform
etCommerce.attachEvent({'mousedown' : 'partialButton'}, 'orderPartialCancellation',
'Bestellnummer 1' , products);

// Neue Darstellungsform mit Array
etCommerce.attachEvent({'mousedown' : ['partialButton', ‘partialButton2’]},
'orderPartialCancellation', 'Bestellnummer 1' , products);
```

9.2 Version 1.1.2

- Die Definition von Varianten im Produkt-Objekt hat sich geändert. In Zukunft lassen sich auch Keys für die Varianten definieren. (Objektdefinition anstatt eines Arrays)

```
// Alte Darstellungsform
...
variants : [‘gelb’, ‘weiblich’, ‘dünn’],
...

// Neue Darstellungsform
...
variants : {'Farbe' : 'gelb', 'Geschlecht' : 'weiblich', 'Figur' : 'dünn'},
…
```

.

9.3 Version 1.1.3

- Typographie-Fehler in Code-Beispielen behoben („onmousedown“ geändert in „mousedown“).