Documentación del API Oct8ne

Versión del API 2.4 (Marzo 2020 / rev04)

oct8ne
fueling eCommerce sales

Índice de contenidos

1. Introducción

La tecnología propietaria de **oct8ne** permite a las compañías proporcionar a sus clientes una experiencia en la compra online similar a la obtenida visitando un comercio físico. Agentes y clientes ven productos y buscan juntos, creando oportunidades para incrementar el consumo gracias a la posibilidad de realizar ventas dirigidas y ventas cruzadas en tiempo real. A diferencia de los servicios de live chat tradicionales, **oct8ne** proporciona a los agentes herramientas para generar ventas mientras se provee de asesoramiento especializado. La inteligencia y analíticas post-visita permiten a las compañías optimizar continuamente sus procesos de venta.

Se trata de un componente ligero, integrable en cualquier plataforma web de comercio electrónico y distribuido siguiendo el modelo SaaS (software as a service), que permite a los comercios introducir proactivamente agentes comerciales durante el proceso de venta con objeto de asistir en tiempo real a los clientes y darles a elegir sus productos, recomendándoles opciones en función de sus necesidades y ofreciéndoles soporte para finalizar con éxito la operación, ofreciendo una experiencia en la compra similar a la obtenida en una tienda física.

El objetivo de este documento es proporcionar a arquitectos, ingenieros de software y desarrolladores información técnica de utilidad para integrar **oct8ne** en su plataforma de comercio electrónico.

Nota: Existen plugins o extensiones oficiales gratuitas para algunas de las principales plataformas de comercio electrónico, proporcionando una integración más sencilla con estos productos sin necesidad de realizar desarrollos personalizados. Antes de comenzar a desarrollar su solución personalizada, consulte las plataformas disponibles en la web del producto www.oct8ne.com.

Para ello, en primer lugar, para ofrecer una visión global del sistema se describirá brevemente la arquitectura de **oct8ne** y dónde encaja cada uno de sus componentes en un escenario de uso habitual.

Seguidamente se realiza una descripción a alto nivel del proceso de instalación de **oct8ne** en plataformas de comercio electrónico, incluyendo los conceptos básicos de la creación de adaptadores personalizados para plataformas no estándar.

Más adelante describiremos en detalle el API, métodos y estructuras de datos usados, que permitirán construir adaptadores para lograr una integración perfecta de **oct8ne** con cualquier plataforma.

Recuerde que para cualquier duda o sugerencia nos tiene a su disposición a través del correo electrónico devsupport@oct8ne.com, donde puede obtener soporte técnico personalizado.

Muchas gracias por usar el API de **oct8ne**.

Happy coding!

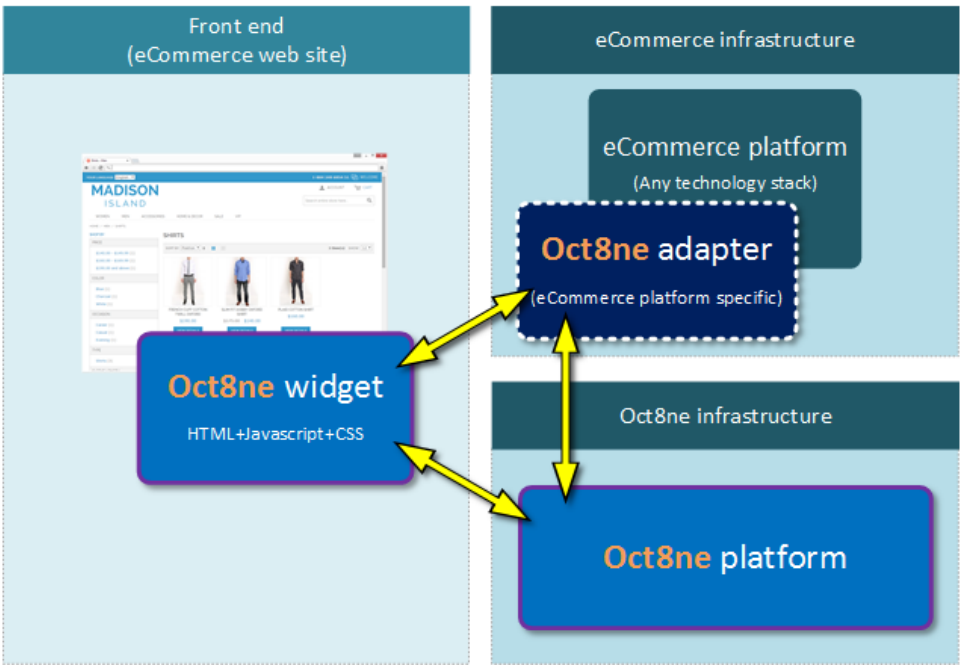
2. Arquitectura

Los principales componentes de **oct8ne** son los siguientes:

- **Oct8ne platform**, compuesto por el conjunto de componentes de servidor alojado sobre la infraestructura de **oct8ne** que proporcionan servicios web y de tiempo real al resto de elementos.
- **Oct8ne widget**, consta de los elementos HTML, CSS y Javascript que, incrustados en su plataforma de comercio electrónico, habilitarán las funcionalidades de **oct8ne**. Más adelante se describe cómo introducir este componente sobre su plataforma.
- **Oct8ne adapter**, es el complemento desarrollado específicamente para la plataforma de comercio electrónico con la que se desea integrar **oct8ne**. En la mayoría de casos utilizará la misma pila tecnológica que la plataforma de eCommerce con la que se integra (.NET, PHP, Java, etc.) y estará alojado en su misma infraestructura.

> **Nota:** Si desea integrar **oct8ne** en su plataforma de eCommerce **deberá desarrollar exclusivamente este componente**, pues el resto son proporcionados por Oct8ne Inc.

El siguiente diagrama representa estos componentes, su ubicación y conexiones entre ellos en el contexto de un sistema de comercio electrónico genérico funcionando con **oct8ne**:

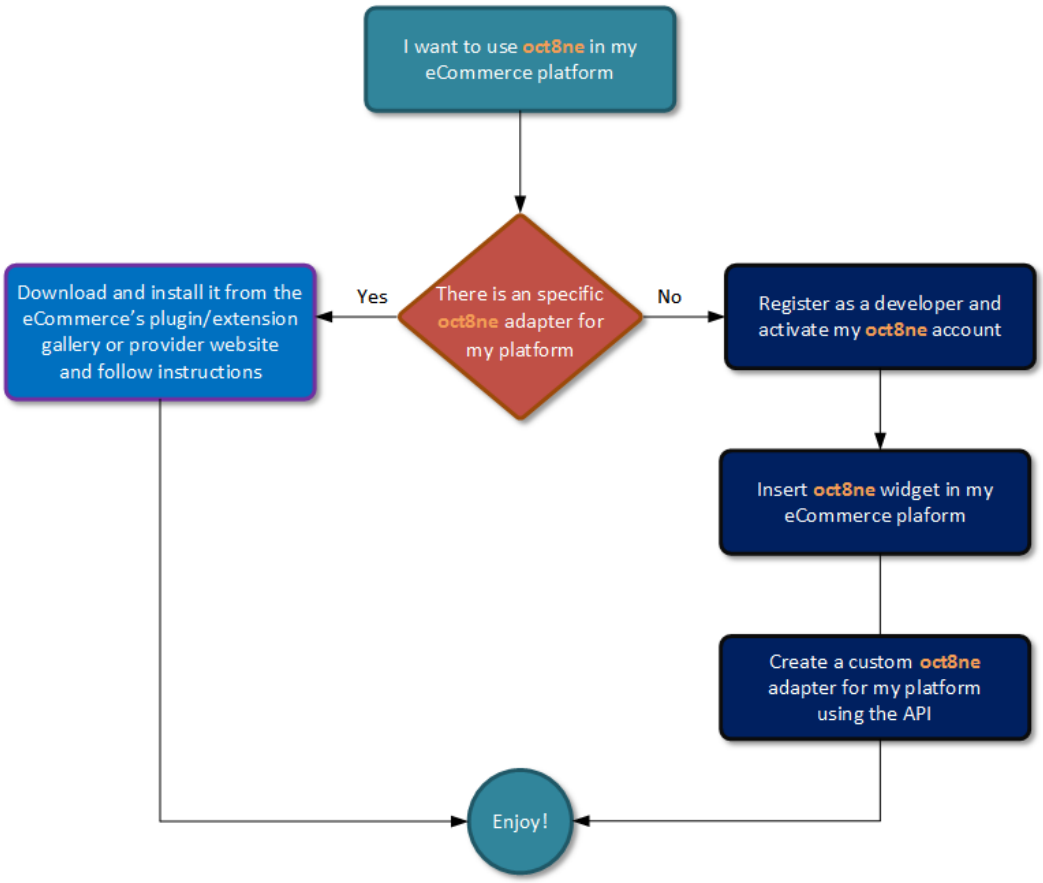


Como se muestra en el diagrama anterior, ni el widget ni la propia plataforma de **oct8ne** interactuarán en ningún momento con su sistema de comercio electrónico de forma directa, lo

hacen siempre a través de la extensión específicamente diseñada para éste. Esta arquitectura hace posible que **oct8ne** sea una solución adaptable a cualquier plataforma de eCommerce, independientemente de la tecnología sobre la que esté construido.

3. Vista general

La integración de **oct8ne** en una plataforma de comercio electrónico se ajustará al siguiente procedimiento:



3.1. Descargar e instalar plugin/extensión

Si su plataforma de eCommerce dispone de adaptadores específicos, podrá encontrarlos normalmente en la galería de plugins o extensiones, o bien en el sitio web de sus desarrolladores, aunque el lugar exacto dependerá de la plataforma. Consulte el manual o la documentación para más información.

Siga las instrucciones facilitadas por el desarrollador de la extensión. Habitualmente el proceso de instalación es sencillo y permite una rápida puesta en marcha de **oct8ne** en su sistema.

3.2. Registro como desarrollador y activación de la plataforma

Para tener acceso al API y a una cuenta de pruebas que le permitirá desarrollar sus adaptadores **oct8ne** necesitará disponer de una cuenta de desarrollador.

Puede solicitarla enviando un correo electrónico a devsupport@oct8ne.com con sus datos de contacto y una breve descripción del proyecto que va a desarrollar, que responderemos

indicando sus credenciales de acceso a la infraestructura de pruebas y toda la información adicional necesaria para integrar **oct8ne** en su plataforma.

3.3. Insertar el widget oct8ne

Para activar los servicios de oct8ne en su plataforma de comercio electrónico debe introducir el siguiente código de script de forma que esté presente en todas las páginas en las que desee que estos servicios estén disponibles. Normalmente esto se puede conseguir introduciendo el script en la página maestra o layout de su sitio web, aunque dependiendo de la plataforma podrían existir otras formas de hacerlo.

```
<script  type="text/javascript">
   var oct8ne = document.createElement("script");
   oct8ne.type = "text/javascript";
   oct8ne.server = "OCT8NE-SERVER-HERE";
   oct8ne.src =
      (document.location.protocol=="https:"?"https://":"http://")
      + [OCT8NE-STATIC-SERVER-HERE]api/v2/oct8ne.js'
      + '?' + (Math.round(new Date().getTime() / 86400000));

   oct8ne.async = true;

   oct8ne.license = "YOUR-LICENSE-HERE";
   oct8ne.baseUrl = "//yourdomain.com";
   oct8ne.checkoutUrl = oct8ne.baseUrl + "/checkout";
   oct8ne.loginUrl = oct8ne.baseUrl + "/login";
   oct8ne.checkoutSuccessUrl = oct8ne.baseUrl + "/checkout/success";
   oct8ne.locale = "en-US";
   oct8ne.currencyCode = "USD";
   oct8ne.apiVersion = "2.4";

   oct8ne.currentProduct = {
       id: "84",
       thumbnail: "http://myserver.com/img/thumbs/84.jpg"
    };

   var s = document.getElementsByTagName("script")[0];
   s.parentNode.insertBefore(oct8ne, s);
</script>
```

Nota: El valor de [OCT8NE-STATIC-SERVER-HERE] es suministrado en el momento de registrarse como desarrollador.

Una buena ubicación para esta porción de script podría ser justo antes de finalizar la etiqueta <BODY> de la página web, aunque en cualquier caso depende de la maquetación de su sitio web.

En este código script es obligatorio establecer el valor de las siguientes propiedades con el valor correcto:

Required properties		
server	string	Dirección del servidor que se le suministró en el momento de registrarse como desarrollador
src	String	URL absoluta hacia el archivo de scripts del API **oct8ne** que le será facilitada en el momento de registrarse como desarrollador. Puede diferir de la que se suministrará en entornos de producción. En el ejemplo de código anterior se añade un parámetro adicional a la URL para forzar la carga de la última versión del script al menos una vez al día.
license	String	Código de licencia que se le suministró en el momento de registrarse como desarrollador.
baseUrl	String	URL absoluta de la raíz de su sitio web.
checkoutUrl	String	URL absoluta de la página de la plataforma de comercio electrónico a la que hay que redirigir al visitante para finalizar la compra.
loginUrl	String	URL absoluta de la página de login de usuario en la plataforma de comercio electrónico.
checkoutSuccessUrl	String	URL de la página mostrada al usuario justo después de finalizar su compra, habitualmente utilizada para agradecer al usuario su confianza y mostrarle información útil para el posterior seguimiento de su compra.
locale	String	Código estándar del idioma/región seleccionado por el cliente del comercio electrónico, por ejemplo “en-US” o “es-ES”.
currencyCode	String	Código estándar de la moneda seleccionada actualmente por el cliente del comercio electrónico, por ejemplo “USD” o “EUR”.
apiVersion	String	Versión de la API implementada por el adaptador. En este documento, se corresponde con la versión: “2.4”
currentProduct	object	Objeto que describe el producto que está siendo visualizado, mediante las propiedades: `id` (identificador del producto) `thumbnail` (URL de la miniatura del producto).

Nota: Debe establecer la propiedad `currentProduct` siempre que la página actual se corresponda a un producto, de forma que **oct8ne** pueda registrar la información del producto. En caso contrario no es necesario definirla.

Existen otras propiedades que pueden ser establecidas, e incluso es recomendable hacerlo, para lograr una mejor integración de **oct8ne** en su plataforma de eCommerce. Las veremos más adelante, en el capítulo "Integración avanzada de oct8ne".

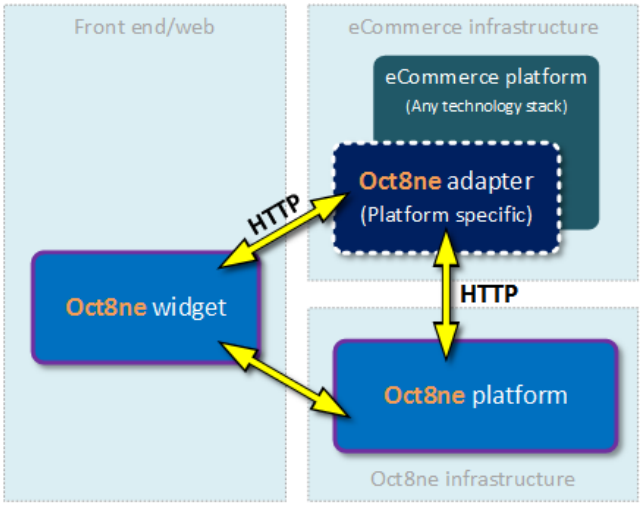
3.4. Crear un adaptador personalizado

Oct8ne ofrece un interfaz de programación de aplicaciones (API) que permite integrar sus servicios de asistencia y colaboración en cualquier tipo de plataforma de terceros mediante la implementación de un adaptador personalizado.

La creación de un adaptador para una plataforma de eCommerce determinada consiste en implementar, usando la tecnología más apropiada en cada caso, un componente alojado en su infraestructura capaz de publicar mediante el protocolo HTTP una serie de métodos definidos por el API de **oct8ne** con el objetivo de intercambiar información con la plataforma para la cual ha sido construido y con el resto de componentes existentes en **oct8ne**, descritos con anterioridad.

Las invocaciones a estos métodos se producen ante determinadas acciones del usuario o del propio sistema. Por ejemplo, cuando el usuario realiza una búsqueda sobre el interfaz de **oct8ne**, el widget establece una comunicación con el adaptador del comercio electrónico, y éste es el que, utilizando los medios de que disponga su plataforma, obtiene y retorna los resultados de vuelta al widget, quien finalmente es quien los muestra al usuario en pantalla.

Otro ejemplo de interacción se produce cuando **oct8ne** necesita conocer los productos relacionados con un producto determinado. En este caso, el widget solicitará al adaptador dicha información, quien la obtendrá desde el sistema de comercio electrónico y la retornará al widget para que la muestre al usuario.



Los adaptadores de **oct8ne** para plataformas de comercio electrónico deben implementar todos los métodos definidos por el API, que serán invocados por **oct8ne**:

Métodos	Componentes involucrados	Momento en que se produce
productInfo	Widget→Adapter	**Oct8ne** necesita obtener todos los detalles de un producto para mostrarlo al usuario.
productSummary	Widget→Adapter	**Oct8ne** necesita obtener un resumen de un producto para mostrarlo al usuario.
search	Widget→Adapter	El usuario busca un producto usando

		el buscador integrado en **oct8ne.**
productRelated	Widget→Adapter	**Oct8ne** necesita conocer los productos relacionados con otro.
customerData	Widget→Adapter	**Oct8ne** necesita refrescar el estado de la sesión del usuario conectado actualmente a la plataforma de eCommerce.
addToWishlist	Widget→Adapter	El usuario añade un elemento a su lista de productos favoritos desde el interfaz de **oct8ne**.
getCart	Widget->Adapter	**Oct8ne** necesita conocer el contenido actual de la cesta de la compra.
getAdapterInfo	Platform→Adapter	La plataforma necesita obtener información sobre el adaptador.
getOrders	Platform→Adapter	Listado de pedidos realizados por el cliente, este listado se le muestra al agente.
getOrderDetails	Platform→Adapter	Detalle de un pedido realizado por el cliente, este detalle se le muestra al agente.

Nota: el sistema podría comportarse de forma indebida si no se implementa alguno de los métodos en el adaptador.

Adicionalmente, existen integraciones avanzadas que permiten conseguir una mejor experiencia de uso de **oct8ne**, cubriendo algunos escenarios más sofisticados. Éstas serán descritas más adelante, en el capítulo “Integración avanzada de oct8ne”.

4. Integración usando oct8ne API

4.1. Notas generales sobre la implementación

En este epígrafe se describen algunos detalles importantes que afectan a todos los métodos y llamadas del API **oct8ne**.

4.1.1. Utilización de JSONP

Para evitar problemas con las llamadas entre orígenes distintos, el API de **oct8ne** utiliza en todos sus puntos de entrada la técnica llamada *JSON with padding*, también conocida como JSONP (http://en.wikipedia.org/wiki/JSONP).

Así, todas las llamadas realizadas desde el widget **oct8ne** hacia el adaptador de su plataforma pueden incluir un parámetro adicional, llamado `callback`, que contiene el nombre de la función que será ejecutada en el lado cliente al recibir la respuesta. Desde su adaptador simplemente deberá retornar el contenido JSON con la siguiente estructura:

```
<callbackFunction>( <jsonData> );
```

Por ejemplo, un método del API del adaptador que reciba como parámetro `callback` el valor `"myFunc"` podría retornar el siguiente contenido, siempre con el content-type establecido como “application/javascript”:

```
myFunc({
    productCount: 0,
    error: false
})
```

Debido a la utilización de JSONP, el widget de **oct8ne** utiliza el verbo HTTP **GET** para realizar consultas al adaptador.

Si el parámetro `callback` no está presente en la petición, los datos deben ser retornados utilizando JSON puro, con content-type “application/json”, por ejemplo:

```
{
    productCount: 0,
    error: false
}
```

4.1.2. Ruta de las llamadas al adaptador

La versión actual del API de **oct8ne** requiere que todos los métodos del adaptador se encuentren en una ruta predeterminada, **/oct8ne/frame/<method>**.

En versiones posteriores del API se permitirá modificarla y adaptarla a su esquema de rutas o necesidades, pero de momento esta posibilidad no está disponible.

4.2. Método "productInfo"

Este método del adaptador es invocado por el widget **oct8ne** cada vez que se necesita obtener la información completa de uno o varios productos. Todos los adaptadores deben implementar este método.

URL	
<yourserver.com>/oct8ne/frame/productInfo	
Parámetros de entrada	
productIds	Identificadores en la plataforma de los productos cuyos datos se quieren obtener, separados por el carácter coma.
locale	Código estándar del idioma en el que deben retornarse los resultados de la consulta, por ejemplo, "es-ES" o "en-US". Si no se indica, se usará el idioma por defecto del sitio web.
currency	Código estándar de la moneda en la que deben obtenerse los precios de los productos, por ejemplo "USD" o "EUR". Si no se indica, se usará la moneda por defecto del sitio web.
Ejemplos de petición	
GET http://www.myserver.com/oct8ne/frame/productInfo?productIds=1&callback=myFunc **GET** http://www.myserver.com/oct8ne/frame/productInfo?productIds=1,2&callback=myFunc **GET** http://www.myserver.com/oct8ne/frame/productInfo?productIds=1&callback=myFunc &locale=en-US&currency=USD	
Ejemplo de respuesta (JSONP)	

```
myFunc([
    <ProductInfo-1>,
    <ProductInfo-2>,
    ...
    <ProductInfo-N>,
])
```

Nota: Recuerde que la respuesta a este método, debe devolver un array de productos con la estructura "ProductInfo", descrita a continuación.

4.2.1. La estructura ProductInfo

Contiene información sobre un producto.

ProductInfo		
internalId	String	Identificador interno del producto en la plataforma eCommerce.
title	String	Nombre o título del producto.

description	String	Texto de descripción ampliada del producto. **Debe ser apropiado para mostrar en una página web** (por ejemplo, los saltos de línea deberían ser convertidos a tags antes de retornarlo).
formattedPrice	String	Precio final del producto. **Debe ser formateado antes de retornarlo al widget** teniendo en cuenta el valor de los parámetros "locale" y "currency" de la consulta, por ejemplo introduciendo puntos para separar los millares, o coma para separar decimales, así como el sufijo o prefijo de la moneda. Si desea que Oct8ne no muestre precios de un producto, simplemente retorne un valor nulo (`null`).
formattedPrevPrice	String	Precio del producto antes de aplicar el descuento. Si el producto no tiene descuento, contiene el mismo valor que la propiedad `formattedPrice`. **Debe ser formateado completamente antes de retornarlo al widget** según los parámetros "locale" y "currency" de la búsqueda.
productUrl	String	URL absoluta hacia la página de detalles del producto, desde donde se puede añadir a la cesta de la compra seleccionando los atributos necesarios (tallas, colores, etc.).

addToCartUrl	String	URL absoluta a la que hay que enviar una petición para añadir el producto a la cesta de la compra. Obligatorio cuando `useProductUrl` vale `false`; en este caso, cuando el visitante añada un producto al carro de la compra usando **oct8ne**, el widget realizará una petición AJAX a esta dirección.
		
		El verbo HTTP a utilizar en la petición AJAX será por defecto "POST". Sin embargo, es posible modificarlo estableciendo la siguiente propiedad en el objeto `oct8ne` del widget:
		
		`<script type="text/javascript">`
		`var oct8ne = document.createElement("script");`
		`...`
		`oct8ne.addToCartHttpVerb = "GET";`
		`...`
		`</script>`

useProductUrl	Boolean	Si vale `true`, indica que el producto debe ser añadido a la cesta de la compra redirigiendo al usuario a la página del producto en el eCommerce. Esto ocurrirá, por ejemplo, si el producto tiene atributos que deben ser seleccionados antes de añadirlos a la cesta de la compra, como tallas o colores. Cuando contiene `false`, el widget realizará la llamada Ajax para añadir el producto a la cesta de la compra, según se describe en la propiedad `addToCartUrl`.

thumbnail	String	URL absoluta hacia la imagen en miniatura del producto. Pueden ser de cualquier tamaño, pero para una óptima visualización se recomiendan imágenes de 120px de ancho y alto.
medias	Array	Array de objetos `Media` con las imágenes del producto. **Como mínimo debe contener un elemento**, que será el usado como imagen por defecto del producto. Las imágenes referenciadas pueden ser de cualquier tamaño, pero para una óptima visualización se recomienda un máximo de 800px de ancho y alto.

El siguiente bloque de código muestra un objeto `ProductInfo`:

```
{
    internalId:             "23",
    title:                  "Product title",
    description:            "This is the <b>product description</b>",
    formattedPrice:         "1,500.00 €",
    formattedPrevPrice:     "2,000.00 €",
    useProductUrl:          true,
    productUrl:             "//myserver.com/cart/add/23",
    addToCartUrl:           "//myserver.com/cart/ajax/add/23",
    thumbnail:              "//myserver.com/thumbs/23.jpg",
    medias: [
       { url: "//myserver.com/images/23-1.jpg" },
       { url: "//myserver.com/images/23-2.jpg" },
       { url: "//myserver.com/images/23-3.jpg" }
    ]
}
```

4.2.2. La estructura Media

Contiene información sobre una imagen o recurso de un producto.

Media		
url	String	URL absoluta hacia la imagen o recurso.

```
{
    url: "//myserver.com/images/23-3.jpg"
}
```

4.3. Método "productSummary"

Este método del adaptador es invocado por el widget **oct8ne** cada vez que se necesita obtener la información resumida de uno o varios productos. Todos los adaptadores deben implementar este método.

URL	
<yourserver.com>/oct8ne/frame/productSummary	
Parámetros de entrada	
productIds	Identificadores en la plataforma de los productos cuyos datos se quieren obtener, separados por el carácter coma.
locale	Código estándar del idioma en el que deben retornarse los resultados de la consulta, por ejemplo, "es-ES" o "en-US". Si no se indica, se usará el idioma por defecto del sitio web.
currency	Código estándar de la moneda en la que deben obtenerse los precios de los productos, por ejemplo "USD" o "EUR". Si no se indica, se usará la moneda por defecto del sitio web.
Ejemplos de petición	
GET http://www.myserver.com/oct8ne/frame/productSummary?productIds=1&callback=myFunc **GET** http://www.myserver.com/oct8ne/frame/productSummary?productIds=1,2&callback=myFunc **GET** http://www.myserver.com/oct8ne/frame/productSummary?productIds=1&callback=myFunc &locale=es-ES&currency=EUR	
Ejemplo de respuesta (JSONP)	

```
myFunc([
    <ProductSummary-1>,
    <ProductSummary-2>,
    ...
    <ProductSummary-N>,
])
```

Nota: Recuerde que la respuesta a este método, debe devolver un array de productos con la estructura "ProductSummary", descrita a continuación.

4.3.1. La estructura ProductSummary

Contiene información resumida sobre un producto.

ProductSummary		
internalId	String	Identificador interno del producto en la plataforma eCommerce.
title	String	Nombre o título del producto.

formattedPrice	String	Precio final del producto. **Debe ser formateado antes de retornarlo al widget** teniendo en cuenta el valor de los parámetros "locale" y "currency" de la consulta, por ejemplo introduciendo puntos para separar los millares, o coma para separar decimales, así como el sufijo o prefijo de la moneda. Si desea que Oct8ne no muestre precios de un producto, simplemente retorne un valor nulo (`null`).
formattedPrevPrice	String	Precio del producto antes de aplicar el descuento. Si el producto no tiene descuento, contiene el mismo valor que la propiedad `formattedPrice`. **Debe ser formateado completamente antes de retornarlo al widget** según los parámetros "locale" y "currency" de la búsqueda.
productUrl	String	URL absoluta hacia la página de detalles del producto, desde donde se puede añadir a la cesta de la compra seleccionando los atributos necesarios (tallas, colores, etc.).
thumbnail	String	URL absoluta hacia la imagen en miniatura del producto. Pueden ser de cualquier tamaño, pero para una óptima visualización se recomiendan imágenes de 120px de ancho y alto.

El siguiente bloque de código muestra un objeto `ProductSummary`:

```
{
    internalId:             "23",
    title:                  "Product title",
    formattedPrice:         "$1,500.00",
    formattedPrevPrice:     "$2,000.00",
    productUrl:             "//myserver.com/cart/add/23",
    thumbnail:              "//myserver.com/thumbs/23.jpg"
}
```

4.4. Método "search"

Este método del adaptador para su plataforma de comercio electrónico es invocado por el widget cada vez que el usuario realiza una búsqueda de productos utilizando alguna de las herramientas proporcionadas por **oct8ne**. Todos los adaptadores deben implementar este método.

URL	
<yourserver.com>/oct8ne/frame/search	
Parámetros de entrada	
search	Texto a buscar.
page	Página de datos a retornar, siendo 1 la primera página.
pageSize	Tamaño de la página. Por defecto, 10.
orderby	Criterio de ordenación. Debe ser uno de los valores siguientes (case sensitive): • relevance (default) • price • name Si su plataforma de comercio electrónico no implementa alguno de los filtros, deberá proporcionar una ordenación razonable a los productos que retorne de la búsqueda. Por ejemplo, si en su plataforma no existe nada similar a "relevance" que permita ordenar los productos, podría retornarlos ordenados por fecha de actualización, por precio, o por otro criterio que considere más acertado en su escenario.
dir	Sentido de la ordenación. Debe ser uno de los siguientes valores (case sensitive): • asc (default) • desc
locale	Código estándar del idioma en el que deben retornarse los resultados de la consulta, por ejemplo, "es-ES" o "en-US". Si no se indica, se usará el idioma por defecto del sitio web.
currency	Código estándar de la moneda en la que deben obtenerse los precios de los productos, por ejemplo "USD" o "EUR". Si no se indica, se usará la moneda por defecto del sitio web.
Ejemplos de petición	
GET http://www.myserver.com/oct8ne/frame/search?search=shoes&page=1&pageSize=10 &callback=myFunc **GET** http://www.myserver.com/oct8ne/frame/search?search=shoes &locale=es-MX&currency=USD&callback=myFunc	
Ejemplo de respuesta (JSONP)	

```
myFunc({
    total: <number of products found>,
    results: [
        <ProductSummary-1>,
        <ProductSummary-2>,
        <ProductSummary-N>,
   ]
   filters: <FilterInfo>
})
```

Además de los parámetros de entrada anteriores, **el widget incluirá en la llamada el valor de los filtros activos**. Por ejemplo, si tenemos un único filtro activo definido con el nombre "color" y el valor "black", la llamada completa a este método podría ser:

myserver.com/oct8ne/frame/search?search=shoes&page=1&pageSize=10&color=black

4.4.1. La estructura FilterInfo

Esta estructura contiene información sobre los filtros aplicados y disponibles en el resultado de una búsqueda.

FilterInfo		
applied	Array	Array de objetos <AppliedFilter>
available	Array	Array de objetos <AvailableFilter>

```
{
    applied: [
        <AppliedFilter-1>
        <AppliedFilter-2>
        <AppliedFilter-N>
    ]
    available: [
        <AvailableFilter-1>
        <AvailableFilter-2>
        <AvailableFilter-N>
    ]
}
```

La propiedad `applied` es opcional, y, si se incluye, debe contener un array de objetos de tipo `AppliedFilter` que describen los filtros que se están aplicando en la consulta realizada.

El tipo **AppliedFilter** contiene datos sobre un filtro aplicado actualmente en la búsqueda.

AppliedFilter		
param	String	Nombre del parámetro enviado al método "Search", por ejemplo "category".

paramLabel	String	Texto a mostrar en el UI para identificar este criterio de filtro. Por ejemplo, “Product category”.
value	String	Valor actual del filtro, por ejemplo “18”.
valueLabel	String	Texto a mostrar que describe el valor actual del filtro. Por ejemplo “Sports”.

El siguiente código representa un filtro aplicado, donde se está indicando que el criterio usado es el campo “category” con un valor de “18”. En pantalla, aparecerá descrito con el texto

```
{
    param: "category",
    paramLabel:  "Product category",
    value: "18",
    valueLabel: "Sports"
}
```

“Sports”:

Por otra parte, la propiedad `available` de la estructura `FilterInfo` también es obligatoria y debe contener un array de objetos `AvailableFilter` que describen los filtros que se pueden aplicar al resultado de la consulta.

El tipo **AvailableFilter** contiene los filtros disponibles tras realizar una búsqueda, permitiendo al usuario afinar los resultados. Cada plataforma puede definir sus propios criterios de filtro.

Visualmente se muestran como una serie de criterios de filtrado agrupados por tipología, junto a los cuales aparece el número de elementos o productos que cumplirían los criterios de la búsqueda si el usuario decide seleccionarlos.

Por ejemplo, en la captura de pantalla se observan los filtros disponibles tras realizar la búsqueda del texto “text”. Aparecen dos grupos de filtros denominados “Category” y “Price range”. En el primero de ellos se pueden observar las distintas categorías en las que existen productos en cuya descripción se encontró el texto “text”, y el número de productos disponibles en cada una de ellas. Más abajo pueden observarse los rangos de precio en los que existen productos.

A continuación se describe de forma detallada esta estructura de datos.

AvailableFilter				
param	String	Nombre del parámetro que será enviado al método “Search” cuando este filtro se active. Por ejemplo, “price”.		
paramLabel	String	Texto que aparece en pantalla, por ejemplo “Price range”.		
options	Array	Colección de objetos `FilterOption` que describen las opciones disponibles en el filtro.		
		FilterOption		
		value	String	Valor del parámetro que será enviado al método “Search” cuando el usuario seleccione esta opción. Por ejemplo, “1”.
		valueLabel	String	El texto a mostrar en el interfaz, por ejemplo “0-99”.
		count	String	Número de productos que cumplen este criterio de búsqueda, teniendo en cuenta los filtros aplicados y la cadena de texto a localizar.

El siguiente bloque de código muestra un objeto de tipo `AvailableFilter`:

```
{
    param:      "price",
    paramLabel: "Price range",
    options: [
        { value: "0", valueLabel: "0-99", count: "1" },
        { value: "2", valueLabel: "200-299", count: "2" },
        { value: "3", valueLabel: "300-399", count: "8" }
    ]
}
```

Nota: la colección de filtros disponibles no debe incluir el filtro activo ni aquellos en los que no existan productos, con el fin de evitar mostrar al visitante opciones de filtro que no tendría sentido seleccionar.

4.5. Método "productRelated"

Este método del adaptador es invocado por el widget **oct8ne** cada vez que se necesitan obtener los productos relacionados con un producto determinado. Todos los adaptadores deben implementar este método.

URL	
<yourserver.com>/oct8ne/frame/productRelated	
Parámetros de entrada	
productId	Identificador en la plataforma del producto cuyos productos relacionados se quieren obtener.
locale	Código estándar del idioma en el que deben retornarse los resultados de la consulta, por ejemplo, "es-ES" o "en-US". Si no se indica, se usará el idioma por defecto del sitio web.
currency	Código estándar de la moneda en la que deben obtenerse los precios de los productos, por ejemplo "USD" o "EUR". Si no se indica, se usará la moneda por defecto del sitio web.
Ejemplos de petición	
GET http://www.myserver.com/oct8ne/frame/productRelated?productId=1&callback=myFunc **GET** http://www.myserver.com/oct8ne/frame/productRelated?productId=1 &locale=en-GB&currency=GBP&callback=myFunc	
Ejemplo de respuesta (JSONP)	

```
myFunc({
    total: <total number of related products found>,
    results: [
        <ProductSummary-1>,
        <ProductSummary-2>,
        <ProductSummary-N>,
   ]
})
```

Nota: debe limitar desde su adaptador el número de productos relacionados retornados por este método. Para optimizar la experiencia del usuario, se estima que entre diez y veinte puede ser un número más que razonable.

Si la plataforma de eCommerce para la que está desarrollando el adaptador no dispone de mecanismos integrados para relacionar productos, puede implementar este método utilizando criterios lógicos de clasificación. Por ejemplo, podría retornar los productos pertenecientes a las mismas categorías que el producto indicado, o que haya sido etiquetado de forma similar. En cualquier caso, estos detalles dependen de las posibilidades de su plataforma de eCommerce.

4.6. Método "customerData"

Este método del adaptador es invocado por el widget **oct8ne** cada vez que se necesita recuperar el estado completo de la sesión del visitante actual, incluyendo el usuario actualmente logado, los elementos presentes en el carro de la compra y los elementos marcados como favoritos por el visitante de la plataforma eCommerce. Todos los adaptadores deben implementar este método.

URL	
<yourserver.com>/oct8ne/frame/customerData	
Parámetros de entrada	
locale	Código estándar del idioma en el que deben retornarse los resultados de la consulta, por ejemplo, "es-ES" o "en-US". Si no se indica, se usará el idioma por defecto del sitio web.
currency	Código estándar de la moneda en la que deben obtenerse los precios de los productos, por ejemplo "USD" o "EUR". Si no se indica, se usará la moneda por defecto del sitio web.
Ejemplos de petición	
GET http://www.myserver.com/oct8ne/frame/customerData?callback=myFunc **GET** http://www.myserver.com/oct8ne/frame/customerData?callback=myFunc &locale=es-MX&currency=USD	
Ejemplo de respuesta (JSONP)	

`myFunc({`
`  <CustomerData>`
`})`	

4.6.1. La estructura CustomerData

Esta estructura contiene el estado completo de la sesión del usuario en **oct8ne**.

CustomerData		
id	String	Identificador del usuario logado en la plataforma. `null` si no ha sido autenticado.
firstName	String	Nombre del usuario logado, o `null` si no fue autenticado.
lastName	String	Apellido del usuario logado, o `null` si no fue autenticado.
email	String	Email del usuario logado, o `null` si no fue autenticado.
wishlist	Array	Array de objetos `ProductSummary`.
cart	Array	Array de objetos `ProductInCart`.

La siguiente porción de código muestra un objeto `CustomerData` que representa a un usuario autenticado que ha añadido un producto a su lista de favoritos y dos a su carro de la compra:

```
{
    id: "1234",
    firstName:  "John",
    lastName:  "Doe",
    email:  "johndoe@myserver.com",
    wishlist: [
       <ProductSummary-2>
    ],
    cart: [
       <ProductInCart-1>
       <ProductInCart-2>
       <ProductInCart-N>
    ]
}
```

4.6.2. La estructura ProductInCart

Esta estructura contiene información útil para el usuario sobre un producto presente en el carro de la compra del visitante de la plataforma eCommerce. Es una especialización de `ProductSummary`, por lo que hereda todas las propiedades de esta estructura y añade las siguientes:

ProductInCart (inherited from ProductInfo)		
qty	Number	Número de elementos añadidos a la cesta de la compra.

La siguiente porción de código muestra un objeto `ProductInCart`:

```
{
    internalId:             "23",
    title:                  "Product title",
    formattedPrice:         "1,500.00 €",
    formattedPrevPrice:     "2,000.00 €",
    productUrl:             "//myserver.com/cart/add/23",
    thumbnail:              "//myserver.com/thumbs/23.jpg"
    qty:                    3
}
```

4.7. Método "addToWishlist"

Este método es invocado por el widget cuando el usuario añade un producto a sus favoritos utilizando las herramientas de **oct8ne**. La implementación de este método es obligatoria si se desea realizar una integración del wishlist con **oct8ne**.

URL	
<yourserver.com>/oct8ne/frame/addToWishList	
Parámetros de entrada	
productIds	Identificadores en la plataforma de los productos a añadir a la lista de favoritos del usuario, separados por comas.
Ejemplos de petición	
GET http://www.myserver.com/oct8ne/frame/addToWishList?productIds=3?callback=myFunc **GET** http://www.myserver.com/oct8ne/frame/addToWishList?productIds=3,8?callback=myFunc	
Ejemplo de respuesta (JSONP)	

`myFunc({`
`  result: "true"`
`})`	

Nota: es responsabilidad del desarrollador del adaptador comprobar si los productos existen previamente en la lista de favoritos, para evitar entradas duplicadas.

4.8. Método "getCart"

Este método es invocado desde el widget para obtener información sobre el contenido de su cesta de la compra. Todos los adaptadores deben implementar este método.

URL
<yourserver.com>/oct8ne/frame/getCart
Parámetros de entrada
Ninguno
Ejemplo de petición
GET http://www.myserver.com/oct8ne/frame/getCart
Ejemplo de respuesta (JSON)
`myFunc({ <CurrentCart> })`

El siguiente bloque de código muestra un ejemplo de la estructura completa retornada en la llamada al método `GetCart`:

```
{
   price: 650.00,
   finalPrice: 703.63,
   currency: "USD",
   totalItems:3,
   cart: [
      {
         internalId: "418",
         title: "Tori Tank",
         qty: 1,
         price: 60.00
      },
      {
         internalId: "339",
         title: "Retro Chic Eyeglasses",
         qty: 2,
         price: 295.00
      }
   ]
}
```

4.8.1. La estructura CurrentCart

Esta estructura contiene información sobre el estado actual de la cesta de la compra del usuario.

CurrentCart		
price	Number	Importe total de la cesta de la compra, sin incluir descuentos, transportes o impuestos. No debe incluir formato ni símbolos de moneda y utilizar el punto como separador decimal, por ejemplo: “1234.56”.
finalPrice	Number	Importe final de la compra, ya incluyendo descuentos, transportes e impuestos. No debe incluir formato ni símbolos de moneda, y utilizar el punto como separador decimal, por ejemplo: “1234.56”.
currency	String	Código estándar de la moneda utilizada actualmente por el cliente del comercio electrónico, por ejemplo “USD” o “EUR”.
totalItems	String	Número total de unidades en la cesta de la compra. Se puede obtener sumando la propiedad `qty` de todos los objetos presentes en la cesta.
cart	Array	Array de objetos `CartItem`.

La siguiente porción de código muestra un objeto `CurrentCart` representando un carro de la compra que contiene dos productos distintos, aunque distintas unidades en cada caso:

```
{
   price: 1000.00,
   finalPrice: 1280.00,
   currency: "USD",
   totalItems: 3,
   cart: [
      <CartItem-1>,
      <CartItem-2>
   ]
}
```

4.8.2. La estructura CartItem

Contiene información sobre un producto que se encuentra actualmente en la cesta de la compra del usuario.

CurrentCart		
internalId	String	Identificador interno del producto en la plataforma eCommerce.
title	String	Nombre o título del producto.
qty	Number	Número de unidades de este producto presentes en la cesta de la compra.

price	Number	Precio unitario del producto. No debe incluir formato ni símbolos de moneda y utilizar el punto como separador decimal, por ejemplo: “1234.56”.

La siguiente porción de código muestra un objeto `CartItem` del que han sido añadidos dos unidades a la cesta de la compra:

```
{
   internalId: "418",
   title: "Tori Tank",
   qty: 2,
   price: 60.00
}
```

4.9. Método "getAdapterInfo"

Este método es invocado tanto desde la plataforma **oct8ne** como desde el lado cliente para obtener información sobre el adaptador. Todos los adaptadores deben implementar este método.

URL
<yourserver.com>/oct8ne/frame/getAdapterInfo
Ejemplos de petición
GET http://www.myserver.com/oct8ne/frame/getAdapterInfo?callback=myFunc **GET** http://www.myserver.com/oct8ne/frame/getAdapterInfo
Ejemplos de respuesta

```
myFunc({
   platform: "Magento",
   adapterName: "Oct8ne official adapter for Magento",
   adapterVersion: "2.5",
   developedBy: "Oct8ne Inc",
   supportUrl: "http://www.oct8ne.com/support/magento",
   apiVersion: "2.4",
   enabled: true
})

{
   platform: "Magento",
   adapterName: "Oct8ne official adapter for Magento",
   adapterVersion: "2.5",
   developedBy: "Oct8ne Inc",
   supportUrl: "http://www.oct8ne.com/support/magento",
   apiVersion: "2.4",
   enabled: true
}
```

A continuación se describen las propiedades de la estructura de datos retornada:

AdapterInfo		
platform	String	Requerido. Plataforma de eCommerce sobre la que corre el adaptador. Puede contener la versión exacta (por ejemplo: "Magento CE 1.9.1").
adapterName	String	Nombre oficial del adaptador.
adapterVersion	String	Requerido. Versión de desarrollo del adaptador.
developedBy	String	Requerido. Desarrollador del adaptador.
supportUrl	String	URL de soporte del adaptador, si existe.
apiVersion	String	Requerido. Versión del API **oct8ne** que utiliza el adaptador. Debe coincidir con la versión del archivo Javascript cargado desde el widget.

enabled	Boolean	Indica si el adaptador está activo en la plataforma de eCommerce, y, por tanto, **oct8ne** está introduciendo su código de script en la página.

4.10. Método "getOrders"

Este método de la plataforma **oct8ne** permite a los desarrolladores de adaptadores dar la posibilidad a los agentes de ver el historial de pedidos realizados por un cliente.

URL	
<yourserver.com>/oct8ne/frame/getOrders	
Parámetros de entrada	
customerEmail	Email del cliente del que se quiere consultar los pedidos.
apiToken	Token de seguridad. Debe compararlo con el token suministrado en el registro de **oct8ne**, y sólo responder a la petición si es correcto.
page	Página a retornar. Por defecto, 1
pageSize	Tamaño de la página. Por defecto, 10.
Locale	Código estándar del idioma en el que deben retornarse los resultados de la consulta, por ejemplo, "es-ES" o "en-US". Si no se indica, se usará el idioma por defecto del sitio web.
currency	Código estándar de la moneda en la que deben obtenerse los precios de los pedidos, por ejemplo "USD" o "EUR". Si no se indica, se usará la moneda por defecto del sitio web.
Ejemplo de petición	

GET
http://www.myserver.com/oct8ne/frame/getOrders
?customerEmail=myemail@web.com
&apiToken=C9562C260EE0B82577D9D315F53EBE78
&page=1
&pageSize=10
&locale=es-ES
&currency=EUR

GET
http://www.myserver.com/oct8ne/frame/getOrders
?customerEmail=myemail2@web.com
&apiToken=C9562C260EE0B82577D9D315F53EBE78
&page=3
&pageSize=10

Ejemplo de respuesta (JSON)

```
[
        {
                "date":"2019-02-28 16:19:15",
                "reference":"VFIYYMABU",
                "total":"55,30\u00a0\u20ac",
                "currency":"EUR",
                "labelState":"En espera de pago por cheque",
                "deliveryDate":""
        },
        {
                "date":"2019-02-28 16:18:10",
                "reference":"EASLDSBCM",
                "total":"170,32\u00a0\u20ac",
                "currency":"EUR",
                "labelState":"Pendiente de envío",
                "deliveryDate":""
        }
]
```

Nota: Si el **apiToken** recibido no es correcto, este método debe retornar un array vacío.

4.11. Método "getOrderDetails"

Este método de la plataforma **oct8ne** permite a los desarrolladores de adaptadores dar la posibilidad a los agentes de ver el detalle de un pedido.

URL	
<yourserver.com>/oct8ne/frame/getOrderDetails	
Parámetros de entrada	
reference	Identificador del pedido
apiToken	Token de seguridad. Debe compararlo con el token suministrado en el registro de **oct8ne**, y sólo responder a la petición si es correcto.
locale	Código estándar del idioma en el que deben retornarse los resultados de la consulta, por ejemplo, "es-ES" o "en-US". Si no se indica, se usará el idioma por defecto del sitio web.
currency	Código estándar de la moneda en la que deben obtenerse los precios del detalle de pedido, por ejemplo "USD" o "EUR". Si no se indica, se usará la moneda por defecto del sitio web.
Ejemplo de petición	

GET http://www.myserver.com/oct8ne/frame/getOrderDetails
?reference=VFIYYMABU
&apiToken=C9562C260EE0B82577D9D315F53EBE78

Ejemplo de respuesta (JSON)


```
{
        "date":"2019-02-28 16:19:15",
        "reference":"VFIYYMABU",
        "total":"55,30\u00a0\u20ac",
        "currency":"EUR",
        "labelState":"En espera de pago por cheque",
        "deliveryDate":"",
        "carrier":"My carrier",
        "trackingNumber":"XXXXX",
        "trackingUrl":"https://mycarrier/orederId/",
        "products":
        [
                {
                "quantity":"3",
                "name":"Brown bear notebook Paper"
                }
        ],
        "comments":
        [
                {
                "message":"Dejarlo en la porter&iacute;a",
                }
        ]
}
```


Nota: Si el **apiToken** recibido no es correcto, este método debe retornar un objeto vacío o un valor nulo.

5. Integración avanzada de oct8ne

En los siguientes epígrafes se describen posibilidades adicionales a la hora de integrar **oct8ne** con su plataforma de comercio electrónico:

- Integración de la tracking bar.
- Integración del Phone PIN.
- Integración de embedded links.
- Actualización en tiempo real del widget **oct8ne**.
- Actualización en tiempo real del eCommerce.
- Link up.

Sugerencia: para una óptima experiencia de uso de **oct8ne** se recomienda implementar todas las integraciones descritas a continuación, si bien no es estrictamente necesario.

5.1. Integración de la history bar

Oct8ne realiza internamente un tracking de los productos visitados por el cliente, y puede mejorar su experiencia de navegación por el catálogo de productos de la plataforma de eCommerce mediante una barra de historial visible en todo momento en la zona superior de la pantalla. En ella aparecerán los últimos productos visitados, con la posibilidad de volver a revisarlos o de acudir a un agente comercial para solucionar cualquier tipo de duda sobre ellos.

Para que esto sea posible, **oct8ne** debe ser consciente de cuándo el visitante se encuentra en la página de detalles de un producto, lo que podemos conseguir estableciendo el valor de la propiedad `currentProduct` del objeto `oct8ne` definido en el script del widget.

Por tanto, en todas las páginas destinadas a ampliar detalles sobre un producto concreto es necesario añadir el siguiente código en el script, asignando a la propiedad `currentProduct` un objeto en el que se indica su identificador interno en la plataforma y la URL absoluta hacia su imagen en miniatura en la propiedad `thumbnail`:

```
<script  type="text/javascript">
    var oct8ne = document.createElement("script");
    ...
    oct8ne.currentProduct = {
       id: "84",
       thumbnail: "http://myserver.com/img/thumbs/84.jpg"
    };
    ...
</script>
```

Si la propiedad `currentProduct` no contiene ningún valor o no está definida, se asumirá que la página actual no está mostrando ningún producto concreto, por lo que no se añadirá a la history bar ninguna información.

La asignación de esta propiedad puede ir envuelta en código de servidor para determinar si se está mostrando un producto, de forma que la asignación sólo se genere en este caso:

```
<?php
   if($currentlyViewingProduct ()) {
      echo 'oct8ne.currentProduct = { id: "'.$currentProductId.'",
           thumbnail: "'.$currentProductThumbnail.'";';
   }
?>
```

5.2. Integración del phone pin

Oct8ne permite a los agentes iniciar muy rápidamente una sesión de asistencia con visitantes que estén contactando con ellos telefónicamente, conociendo el contexto y detalles de la navegación de estos posibles clientes, con el fin de poder proporcionar una mejor ayuda.

Para ello, el visitante deberá suministrar al agente un código PIN que podrá localizar en la página web en la que se encuentran.

El PIN puede ser ubicado en cualquier zona del sitio web, sobre cualquier elemento HTML, por lo que es totalmente personalizable. **Oct8ne** simplemente buscará en la página elementos con la clase CSS “oct8ne-phone” y los configurará para que al pulsar sobre ellos aparezca el PIN justo debajo.

Por ejemplo, el siguiente código muestra cómo se podría mostrar el PIN bajo una imagen de un teléfono alineado a la derecha de la página:

```
<body>
   ...
   <img src="http://www.yourdomain.com/images/phone.png"
        class="oct8ne-phone"
        style="float: right; margin: 0px 0px 9px 9px;
        cursor: pointer;">
   ...
</body>
```

Obviamente, si no se desea aplicar estilos inline también podrían usarse CSS independientes:

HTML:

```
<body>
   ...
   <img src="http://www.yourdomain.com/images/phone.png"
        class="oct8ne-phone">
   ...
</body>
```

CSS:

```
.oct8ne-phone {
   float: right;
   cursor: pointer;
}
```

5.3. Notificación de sesión con pedido realizado

Oct8ne permite saber si una sesión de chat con un cliente, ha finalizado con una venta.

Para ello, se deberá realizar una petición hacia los servidores de Oct8ne mediante la introducción de un script en la página del comercio donde se sepa con certeza que se ha

finalizado una venta (página de confirmación de pedido, etc…). El script ha de contener el id **"oct8ne-sale-notification".**

URL	
https://[OCT8NE-STATIC-SERVER-HERE]api/source/js/ext/sale-notification.js?license='[YOUR-LICENSE-HERE]'&currencyCode='CURRENCY_ISO_CODE'&locale='LOCALE'&value='CART_VALUE'&reference='ORDER_ID'&customerId='CUSTOMER_ID'&contextInfo='CONTEXT_INFO'	
Parámetros de entrada	
license	Código de licencia que se le suministró en el momento de registrarse como desarrollador.
currencyCode	Código estándar de la moneda en la que deben obtenerse los precios del detalle de pedido, por ejemplo “USD” o “EUR”. Si no se indica, se usará la moneda por defecto del sitio web.
locale	Código estándar del idioma en el que deben retornarse los resultados de la consulta, por ejemplo, “es-ES” o “en-US”. Si no se indica, se usará el idioma por defecto del sitio web.
value	Valor final del pedido realizado. Ejemplo: "1237,59 €"
reference	Identificador del pedido en el comercio.
customerId	Identificador del cliente en el comercio.
contextInfo	Información adicional que se quiera guardar. Se trata de un objeto serializado como JSON con pares clave/valor arbitrarias Ejemplo: "{'key1':'value1','key2':'value2'}"
Ejemplo de petición	
https://static.oct8ne.com/api/source/js/ext/sale-notification.js ?license=75CDF90D2A40A53A2899175156FE99E4 &currencyCode=EUR &locale=es-ES &value=589,32 € &reference=KHWLILZLL &customerId=382442 &contextInfo={'key1':'value1','key2':'value2'}	
Ejemplo de script	
<script id="**oct8ne-sale-notification**" type="text/javascript" src=https://static.oct8ne.com/api/source/js/ext/sale-notification.js ?license=C95532140EE0B82566D9D315F53EBE78&CurrencyCode=EUR&locale=es-ES&value=589,32 € &reference=KHWLILZLL&customerId=382442&contextInfo=%7B%27key1%27%3A%27value1%27%2C%27key2%27%3A%27value2%27%7D'> </script>	

Nota: El único campo obligatorio es "license" y el script ha de contener el id "oct8ne-sale-notification".

5.4. Integración de embedded links

El desarrollador de una plataforma de comercio electrónico puede introducir enlaces personalizados hacia distintos servicios proporcionados por **oct8ne**, lo que permite una gran flexibilidad a la hora de añadir elementos visuales atractivos que animen a los visitantes a ponerse en contacto con los agentes comerciales para que les asesoren en sus compras.

Cuando hay agentes conectados, **oct8ne** hace visibles todos los elementos del DOM marcados con la clase CSS “oct8ne-widget-on”. No existe limitación en cuanto al tipo de elemento HTML, o a su ubicación en el layout de la página; puede aplicar esta clase a imágenes, bloques HTML como `DIV` o `SPAN`, o cualquier otro tipo de etiqueta, y posicionarlos en el lugar en el que mejor encaje en su sitio web. La siguiente porción de código muestra un elemento de este tipo y su personalización usando estilos en línea:

```
<body>
   ...
   <div class="oct8ne-widget-on"
        style="display: none;
               position: fixed; top: 0; right: 0; z-index: 9999">
      Agents available, chat now!
   </div>
   ...
</body>
```

Cuando el visitante pulse sobre el elemento, se abrirá el visor **oct8ne** y se iniciará una sesión de chat con el agente que esté disponible en ese momento.

Nota: asegúrese mediante estilos en línea o en hojas CSS que estos elementos son inicialmente invisibles en la página, por ejemplo estableciendo su estilo a `display: none`. **Oct8ne** los mostrará sólo si detecta agentes conectados en la plataforma.

De la misma forma, **oct8ne** mostrará todos los elementos en los que aparezca la clase CSS “oct8ne-widget-off” cuando no existan agentes conectados. En el siguiente bloque de código se muestra un ejemplo de implementación de este enlace en la página, y su personalización mediante hoja de estilos:

HTML:

```
<body>
```

```
   ...
   <div class="oct8ne-widget-off" id="no-agents">
       <p>Contact us
          <small>We'll call you asap</small>
       </p>
   </div>
   ...
</body>
```

CSS:

```
#no-agents {
   display: none;
   position: fixed;
   z-index: 9999;
   right: 0;
   top: 0;
 }
```

Nota: asegúrese mediante estilos en línea o en hojas CSS que estos elementos son inicialmente invisibles en la página, por ejemplo estableciendo su estilo a `display: none`. **Oct8ne** los mostrará sólo si no detecta agentes conectados en la plataforma.

Por defecto, si un visitante pulsa sobre un elemento con la clase "oct8ne-widget-off", se abrirá el formulario de contacto de **oct8ne**. Si se desea modificar este comportamiento, es posible añadir la clase "oct8ne-widget-disabled" e introducir en el elemento un link personalizado, como en el siguiente ejemplo:

```
<div class="oct8ne-widget-off oct8ne-widget-disabled">
     style="display: none; cursor: pointer;"
   <a href="/contact">Contact us</a>
</div>
```

Introduciendo el código anterior, cuando no haya agentes disponibles aparecerá en la página el link "Contact us", y cuando los visitantes pulsen sobre él, serán redirigidos hacia la página "/contact" del sitio web, en lugar de abrirse el formulario de contacto de **oct8ne**.

Obviamente, en la misma página pueden coexistir elementos que utilicen ambas clases con objeto de mostrar un mensaje u otro en función de la disponibilidad de agentes:

```
<body>
   ...
   <div class="oct8ne-widget-on" style="display: none">
      Agents available, chat now!
   </div>
```

```
   <div class="oct8ne-widget-off" style="display: none">
      Contact us!
   </div>
   ...
</body>
```

O incluso pueden ser combinadas en un mismo elemento para hacer que éste sea visible haya o no agentes, con el mismo texto y características:

```
<body>
   ...
   <div class="oct8ne-widget-on oct8ne-widget-off">
     Talk to us!
   </div>
   ...
</body>
```

5.5. Actualización en tiempo real del widget oct8ne

A veces el visitante puede realizar acciones sobre la plataforma de eCommerce que conviene notificar al widget **oct8ne** para que actualice su UI y mantenga su estado sincronizado.

Esto puede ocurrir, por ejemplo, cuando el usuario se identifica como cliente de la plataforma, momento en que es interesante que el widget sea informado para ofrecer servicios más personalizados. O si el usuario utiliza un botón Ajax de la plataforma eCommerce para añadir un producto al carro de la compra o a su selección de favoritos: también sería necesario notificar a **oct8ne** para que su UI muestre el nuevo estado correctamente.

Nota: **oct8ne** recarga el estado completo del widget en cada cambio de página llamando al método “customerData”, por lo que el estado se mantendrá actualizado en la mayoría de ocasiones. Lo descrito en este epígrafe puede ser interesante si el UI de su plataforma de eCommerce utiliza Ajax para operaciones que modifican el estado del sistema sin cambiar de página.

Para estos escenarios, **oct8ne** proporciona un mecanismo de actualización en tiempo real que puede invocarse en cualquier momento desde el adaptador realizando una petición HTTP a la plataforma **oct8ne**. Esta petición debería ejecutarse desde el adaptador de la plataforma eCommerce cuando una las acciones de cambio de estado tengan lugar, lo cual provocará que el widget se actualice.

URL
http://backoffice.oct8ne.com/platformConnection/update

Parámetros de entrada	
visitor	Identificador de la sesión **oct8ne** del usuario, disponible en la cookie "oct8ne-visitor".
what	Qué se desea actualizar, a elegir entre "cart", "wishlist" o "user".
Ejemplo de petición	
POST http://backoffice.oct8ne.com/platformConnection/update?visitor=3847&what=cart	
Ejemplo de respuesta	
Este método no devuelve ningún valor.	

Internamente, la invocación de este método utilizará la infraestructura en tiempo real de **oct8ne** para notificar al widget que su estado debe ser actualizado. El widget, al recibir dicha notificación, invocará al adaptador para obtener en nuevo estado y actualizará su UI apropiadamente.

Es importante tener en cuenta que **esta actualización sólo se producirá cuando exista una conexión de tiempo real entre el visitante y la plataforma oct8ne**, y esto ocurre únicamente cuando el visitante ha entrado en comunicación con un agente comercial de la tienda.

Nota: puede detectar fácilmente si el agente y el visitante están en comunicación, pues en estos casos la cookie "oct8ne-visitor" siempre contiene una cadena de caracteres que identifica la conexión. Por tanto, **si esta cookie no contiene ningún valor, no es necesario que realice la petición**.

En otros casos, es decir, cuando no exista la conexión de tiempo real abierta, la actualización podría forzarse desde el lado cliente llamado al método `update()` del objeto `oct8ne`, como en el siguiente ejemplo:

```
function ajaxAddToCart(productId) {
    $.ajax(                          // Adds the product to the cart
        ...
     )
     .done(function(result) {        // If the operation succeed,
        oct8ne.update("cart");       // refresh oct8ne widget status
     });
}
```

El método `update()` admite un parámetro mediante el cual es posible indicar qué información del widget hay que actualizar. Los valores permitidos son idénticos a los del parámetro `what` de la llamada HTTP vista anteriormente.

5.6. Actualización en tiempo real del eCommerce

De forma similar al caso descrito en el epígrafe anterior, hay veces que el visitante realiza acciones utilizando **oct8ne** que es conveniente notificar a la plataforma eCommerce para que actualice su UI y mantener el estado sincronizado entre ambos sistemas.

Este caso puede aparecer, por ejemplo, en plataformas eCommerce en las que está siempre visible el contenido de la cesta de la compra. En este escenario, si el visitante añade un producto al carro utilizando el botón disponible el widget **oct8ne**, el contenido de la cesta del eCommerce no se actualizará hasta que se refresque o se cambie de página.

Para lograr una mejor integración en estos escenarios, **oct8ne** ofrece un mecanismo de callbacks mediante el cual el desarrollador del adaptador puede tomar el control cuando el usuario realiza estas acciones usando el widget y asegurar la sincronización del interfaz del eCommerce con el nuevo estado del sistema, por ejemplo actualizando vía Ajax el contenido de la cesta de la compra en la página.

Los callbacks están disponibles en el objeto **oct8ne** introducido en las páginas, y pueden ser utilizados como se muestra en la siguiente porción de código:

```
<script  type="text/javascript">
    var oct8ne = document.createElement("script");
    ...
    oct8ne.onProductAddedToCart = function(productId) {
        // The product has been added to the cart using oct8ne
        // Update the ecommerce's UI here
    };
    oct8ne.onProductAddedToWishList = function(productId) {
        // The product has been added to the wishlist using oct8ne
        // Update the ecommerce's UI here
    };
    ...
</script>
```

Nota: tenga en cuenta que estos callbacks son invocados por el widget **una vez que el producto ha sido añadido** a la cesta de la compra o a la lista de deseos, por lo que en su interior no debe volver a añadir estos productos de nuevo.

5.7. Link up

Este método de la plataforma **oct8ne** permite a los desarrolladores de adaptadores automatizar el proceso de registro y obtención del identificador de licencia y token de seguridad del API. Es especialmente interesante si se pretende construir un plugin o extensión para un sistema estándar y proveer a sus gestores un mecanismo de instalación automático, normalmente ejecutado tras la descarga del componente desde una galería de extensiones.

Nota: implemente este método sólo si está creando un plugin o extensión para una plataforma estándar y desea incluir en él un proceso de auto-instalación.

En estos casos, el link up será un paso que necesariamente debe ser realizado antes de poder utilizar **oct8ne** en la plataforma.

URL	
https://backoffice.oct8ne.com/platformConnection/linkup	
Parámetros de entrada	
email	Email con el que ha sido registrada la cuenta **oct8ne** del usuario.
pass	Password de acceso del usuario a **oct8ne**.
platform	Nombre de la plataforma, como “Magento”, “WooCommerce”. En caso de ser un desarrollo personalizado, utilizar “Custom”. Consulte el soporte a desarrolladores para otros casos.
urlDomain	URL base completa del sistema de eCommerce, incluyendo protocolo. Por ejemplo, “http://www.mycommerce.com”.
statusPlatform	Valor lógico (`"true"` o `"false"`) que indica si **oct8ne** está activado en la página en el momento de realizar el linkup.
Ejemplo de petición	
POST https://backoffice.oct8ne.com/platformConnection/linkup content-type: application/x-www-form-urlencoded email=johndoe@gmail.com&pass=mysecurepass&platform=woocommerce &urlDomain=http://www.myecommerce.com&statusPlatform=false	
Ejemplo de respuesta (JSON)	

```
{
    apiToken:  "61958072EB8B476EBF5D6F77F66384AC",
    licenseId: "3BDFF5329C414EBCB116CCE1CE1AD807",
     server:    "subdomain.domain.com/",
     urlStatic: "otrosubdomain.domain.com/"

}
```

6. Soporte

A través de la dirección de correo electrónico devsupport@oct8ne.com estaremos encantados de recibir sus sugerencias y comentarios. También puede utilizar esta dirección de correo para notificar bugs u obtener soporte técnico personalizado.

7. Historial de versiones

7.1. Versión 2.4

A continuación se detallan los cambios introducidos respecto a la versión anterior del API.

- Añadido el nuevo método `GetOrders` al API.
- Añadido el nuevo método `GetOrderDetails` al API.
- Eliminada la sección relativa a la configuración de la "búsqueda asistida"
- Añadida la variable obligatoria `apiVersion` al código de script de la página.
- Añadida la notificación de sesión con pedido realizado.

Nota: si está actualizando su adaptador desde versiones anteriores, recuerde establecer la versión del API retornada en el método `GetAdapterInfo` al valor "2.4".

7.2. Versión 2.3

A continuación se detallan los cambios introducidos respecto a la versión anterior del API.

- Añadida la variable obligatoria `checkoutSuccessUrl` al código de script de la página.
- Añadido el nuevo método `GetCart` al API.
- Añadido en rev03: El archivo principal Javascript se descarga ahora desde CDN "static.oct8ne.com".

Nota: si está actualizando su adaptador desde versiones anteriores, recuerde establecer la versión del API retornada en el método `GetAdapterInfo` al valor "2.3".

7.3. Versión 2.2

A continuación se detallan los cambios introducidos respecto la versión anterior del API.

- Añadidas las variables obligatorias `currencyCode` y `locale` en el código de script de la página para indicar la moneda e idioma que está siendo usado actualmente.
- Los métodos `productInfo`, `productSummary`, `search`, `productRelated` y `customerData` aceptan ahora los parámetros "currency" y "locale" para indicar la moneda y el idioma y región de la consulta.
- En las estructuras `ProductInfo` y `ProductSummary` se han eliminado los campos `price` y `prevPrice`.
- En la estructura `ProductInfo` se ha eliminado el campo `rating`.

- Las estructuras `ProductInfo` y `ProductSummary` retornan ahora los datos de precios completamente formateados, atendiendo a la moneda y cultura actual, en los nuevos campos `formattedPrice` y `formattedPrevPrice`.

Nota: si está actualizando su adaptador desde versiones anteriores, recuerde establecer la versión del API retornada en el método `GetAdapterInfo` al valor "2.2".

7.4. Versión 2.1

A continuación se detallan los cambios introducidos respecto versiones anteriores del API.

- El método `Search` retorna ahora objetos de tipo `ProductSummary`.
- El método `GetAdapterInfo` ahora incluye el campo `enabled`, que permite determinar si el adaptador está activo o inactivo en la plataforma.
- Todos los métodos retornan datos en formato JSON o JSONP dependiendo de la existencia del parámetro `callback`.
- Se ha eliminado el método `GetReportData`.
- Se ha introducido el método `GetSalesReport` para la obtención de datos de ventas.
- Se ha introducido el método `ProductSummary` para obtener datos resumidos de productos.
- Los métodos `CustomerData` y `ProductRelated` ahora usan la estructura de datos `ProductSummary`.
- El nombre de la cookie "TokenVisitor" ha sido cambiado a "oct8ne-visitor".