

# Carousel Integration

by Noesis Srl  
<http://www.noesis-research.com>



CAROUSEL

# Table of Contents

*Carousel as Web Service* .....3

*Carousel as a standalone ad server* .....6

AdJug - Carousel Integration

CAROUSEL



## Carousel as web service

The Ad Server uses the Carousel engine through a Web Service: the Ad Server queries Carousel every time it needs the updated rules to serve ads, and forwards the feedback regarding the performance data.

Carousel needs to know the characteristics of the context (campaign and ads, sites and pages, geographical areas, time slots); for this reason we expose some additional functions in the web service to allow to set this data. Carousel automatically detect if someone changed the context, so, if there is something new.

Below we describe the basic set of configuration functions; however, any additional available information can be used in Carousel with similar functions.

- Supposing that an user is identified only through his/her geographical location derived from the IP address of the request, you have to define the possible values of them.  
Using the method *AddLocation(string name)* it's possible to add various *Location*.
- Each user can run more sites, each one characterized by one or more *Page*. Using the method *AddSite(string name, string url)* it's possible to add various *Site* and using the method *AddPage(string name, string category, int siteId)* it's possible to add various *Page* to the identified site through the ID by parameter.
- If requests could be diversified depending on the particular time of the day, you have to define the different time slots.  
Using the method *AddTimeSlot(string name, int fromTime, int toTime)* it's possible to add various *TimeSlot*.
- Supposing that the ads are grouped in campaigns, you have to define the campaign and for each of them the list of ads involved.  
Using the method *AddCampaign(string name, DateTime start, DateTime end)* it's possible to add various *Campaign* and using the method *AddAdvertisement(string name, string category, int campaignId)* it's possible to add an *Advertisement* to a given campaign identified by means of parameter.

After the configuration steps, the Ad Server can begin a permanent communication process with Carousel, to receive the instructions to optimize the campaign in real time. This communication process is a sequence of calls alternate to other two methods of the web service:

- with *GetRules(DateTime lastTimestamp)* the Ad Server requests the list of all the rules updated since the date indicated by the parameter and receives a list of *rules* indicating which ads show for which user in a specified time slot.  
Timestamp represents the creation date of the rules and is necessary to understand if the information has been updated. If there isn't any new rules, the method gives the precedent list not modified.



- The feedback data represents what has happened on the sites where Carousel's suggestions have been applied; in particular, it is concerned with the number of impressions and clicks received by the ads selected. This data is of fundamental importance because it is by actually using it that the algorithm learns, evolves and generates ever more suitable and optimal rules; because of this, the more frequently that feedback is sent, the easier it will be for Carousel to improve the suggested rules.

Using *SetFeedback(int locationId, int timeSlotId, int adId, int clicks, int impressions)* you can provide to Carousel the performance data needed for the optimization process (how many click and impression the ads obtain for a specified user during a specified time slot).

N.B. Carousel was designed to handle pay-per-action models: the number of clicks obtained is only a particular case of value. Thus the parameter *clicks* can be substituted by a parameter *value*. What is *value* has to be defined for each payment model.

Below you can find a simply schema of the Carousel web service.

- **AddLocation**  
Creates a new geographical location.  
  
**Parameters**  
*name* is the name of the new location.  
  
**Response**  
*The identifier of the new location.*
- **AddTimeSlot**  
Creates a new time slot of the day.  
  
**Parameters**  
*name* is the name of the new time slot.  
*fromHour* is the starting hour of the time slot.  
*toHour* is the ending hour of the time slot.  
  
**Response**  
*The identifier of the new time slot.*
- **AddCampaign**  
Creates a new campaign.  
  
**Parameters**  
*name* is the name of the new campaign.  
*start* is the starting time of the campaign.  
*end* is the ending time of the campaign.  
  
**Response**  
*The identifier of the new campaign.*





- **AddAdvertisement**  
Creates a new advertisement for the specified campaign.  
**Parameters**  
*name* is the name of the new advertisement.  
*category* is the category/group of the advertisement.  
*campaignId* is the identifier of the campaign.  
**Response**  
The identifier of the new advertisement.
- **AddSite**  
Creates a new site.  
**Parameters**  
*name* is the name of the new site.  
*url* is the url address of the site.  
**Response**  
The identifier of the new site.
- **AddPage**  
Creates a new page for the specified site.  
**Parameters**  
*name* is the name of the new page.  
*category* is the category/group of the page.  
*siteId* is the identifier of the site.  
**Response**  
The identifier of the new page.
- **GetRules**  
Returns the array of Rule updated since lastTimeStamp.  
**Parameters**  
*lastTimeStamp* last time of Rules received by the user. This field is mandatory. First time it must be null to indicate that is the first time that the method is called.  
**Response**  
The array of Matching Rule elements.
- **SetFeedback**  
Provide performance data to Carousel.  
**Parameters**  
*locationId* the identifier of the geographical location of the user.  
*timeSlotId* the identifier of the time slot of the day.  
*adId* the identifier of the ad.  
*clicks* the number of clicks obtained.  
*impressions* the number of impression obtained.  
**Response**  
(none)

## Carousel as a stand-alone ad server

It seems particularly suitable for a pilot project because it's based on the ad tag use.

Ad tag could be included in publisher site.

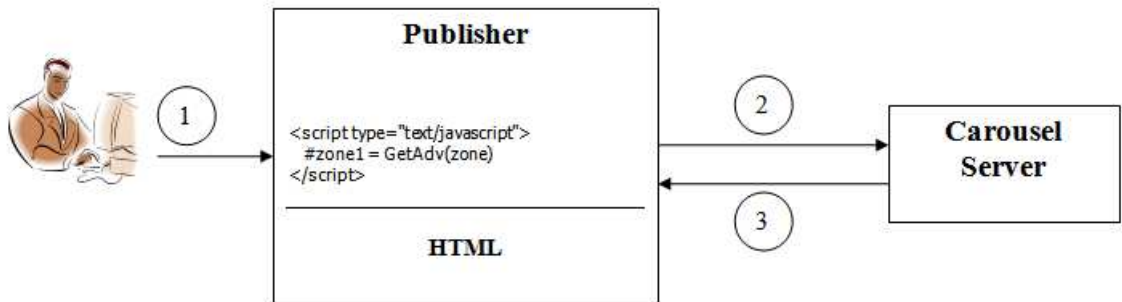


Figura 1 Stand-alone ad server interaction – first case

In this first case:

1. The user asks for a web site page, the browser downloads it and executes the ad tag code.
2. The ad tag asks the Carousel Server for the advertisement.
3. Carousel sends the browser the HTML code containing the URL of the selected advertisement.

Otherwise, the publisher site includes only the URL from which download the Carousel ad tag.

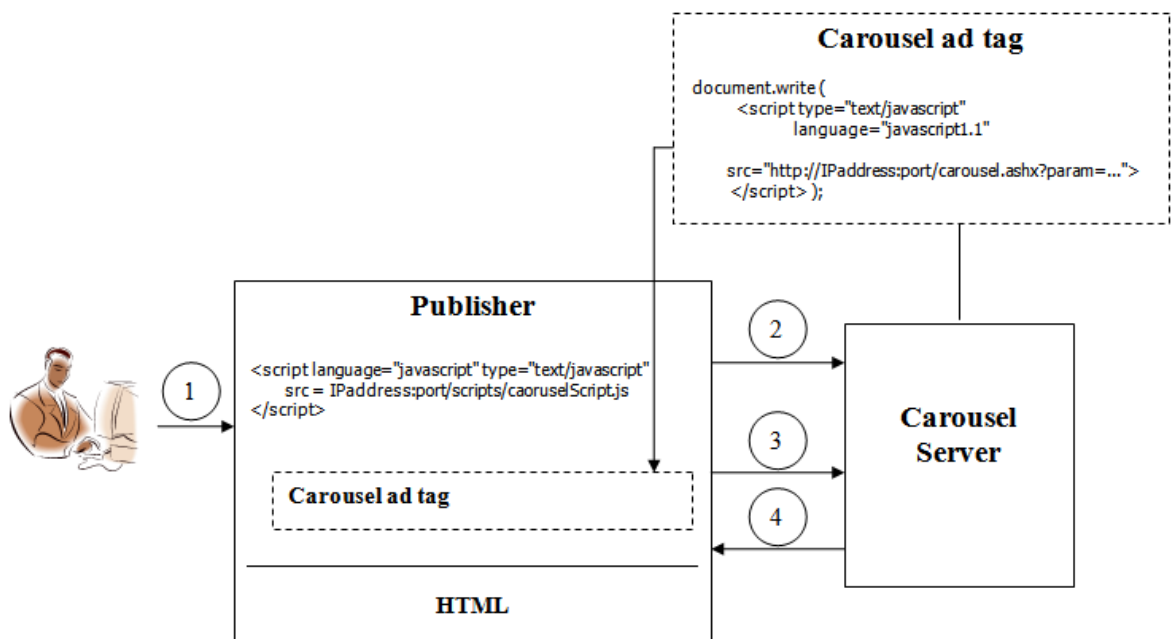


Figura 2 Stand-alone ad server interaction – second case



In this second case:

1. The user asks for a web site page and the browser queries the Ad Server.
2. Ad Server returns a Carousel ad tag containing Carousel URL.
3. The browser calls Carousel by the URL contained in the ad tag attaching parameters to recover the advertisement.
4. Carousel sends the browser the HTML code containing the URL of the selected advertisement.



For additional information, please contact:

**Noesis s.r.l.**

Corso Italia, 89  
I56125 Pisa  
Italy

Telephone +39 050 9911080  
Fax +39 050 9911588  
Email [info@noesis-research.com](mailto:info@noesis-research.com)

[www.noesis-research.com](http://www.noesis-research.com)

*Adjug - Carousel Integration*

# CAROUSEL

