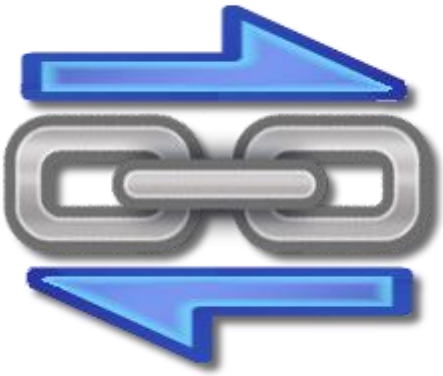
# Linkback Extension



*Pingback & TrackBack*

*Clients and Servers*

*for the Yii PHP Framework*

*Developed for the Yii Community by*

*PBM Web Development*

PBM

# Linkback Extension

## Contents


Introduction ..... 4

Features ..... 4

Definitions ..... 4

Linkback Overview ..... 4

- Ping Information ..... 5
  - Pingback ..... 5
  - TrackBack ..... 5

License ..... 6

Compatibility ..... 6

Installation ..... 6

Configuration ..... 7

- Linkback Server ..... 7
  - LinkbackAutoDiscovery ..... 7
  - LinkbackController ..... 8
- Linkback Client ..... 8
  - LinkbackClientBehavior ..... 8

Linkback Server ..... 10

- LinkbackAutoDiscovery ..... 10
  - Public Properties ..... 10
  - Public methods ..... 11
  - Protected methods ..... 11
  - Property Details ..... 11
  - Method Details ..... 12
- LinkbackController ..... 13
  - Public Properties ..... 14
  - Public methods ..... 15
  - Protected methods ..... 15
  - Events ..... 15
  - Property Details ..... 15
- Method Details ..... 19
- Event Details ..... 20
- Response Codes ..... 21

Linkback Client ..... 22
LinkbackClientBehavior ..... 22
Public Properties ..... 22
Public methods ..... 23
Protected methods ..... 23
Events ..... 23
Property Details ..... 23
Method Details ..... 26
Event Details ..... 26
Appendix A – Linkback Client Database Table Schema ..... 28
Linkbacks Table ..... 28
LinkbackPings Table ..... 28

## Figures

Figure 1 - Example LinkbackAutoDiscover Configuration ..... 7
Figure 2 - Example LinkbackController Configuration ..... 8
Figure 3 - Example LinkbackClientBehavior Configuration ..... 9
Figure 4 - Example LinkbackClientBehavior::linkback() Method Call ..... 9

# Linkback Extension

## Introduction

Linkbacks (Pingbacks and TrackBacks) are a method by which authors can obtain notification when other authors link to one of their documents, enabling them to track who is linking to them, and to provide reciprocal links to the referring document.

The Linkback Extension provides a simple means for blogs to support both Pingbacks and TrackBacks. It implements Pingback and TrackBack clients and servers, and inserts Pingback and TrackBack auto-discovery code into blog posts.

## Features

- Pingback 1.0 compliant client and server
- TrackBack 1.2 compliant client and server
- Automatic server auto-discovery code generation
- Blog system integration
- Simple installation and configuration

## Definitions

The terms Linkback, Pingback, and TrackBack are often used colloquially and interchangeably. In this document the following definitions are used:

| Term | Definition |
|---|---|
| **Linkback** | Pingback and/or TrackBack (see http://en.wikipedia.org/wiki/Linkback) |
| **Pingback** | As defined at http://www.hixie.ch/specs/pingback/pingback |
| **TrackBack** | As defined at http://www.sixapart.com/pronet/docs/trackback_spec |
| **Auto-discovery** | Code in the target document that provides the server's URL to the client. |
| **Ping** | Notification by a Linkback client to a Linkback server. |

## Linkback Overview

While the underlying mechanisms for Pingbacks and TrackBacks differ (Pingbacks use an XML-RPC call (client) and function (server), TrackBacks use a REST model), they both provide the same service, i.e. notification to a document's author about a link by another author to the document, and the process for both is similar.

In the description below, Alice's system is the client, Bob's the server.

1. Alice writes a post on her blog that contains a link to a post on Bob's blog. The permalink to Alice's post is `http://alice.example.com/posts/all-about-linkbacks`; the permalink of Bob's post is `http://bob.example.net/linkbacks`.
2. Alice's blogging system parses the post and extracts all the external links. For each external link, Alice's blogging system does the following:

3. Requests the page referred to by the link. Typically the amount of the page is limited to a few kilobytes.
4. Looks for Linkback auto-discovery code in the page. If no auto-discovery code is found Alice's system goes to step 3 using the next link it found in step 2.
5. Extracts the Linkback server URL from the auto-discovery code and pings it; the ping containing information about Alice's post.
6. Bob's system receives the ping sent by Alice's system.
7. Bob's system confirms that `http://bob.example.net/linkbacks` is a post on his blog.
8. Bob's system then requests the content of Alice's post, `http://alice.example.com/posts/all-about-linkbacks` , to:
    a. confirm it exists,
    b. is text,
    c. contains a link to `http://bob.example.net/linkbacks.`
9. It then extracts any other information about Alice's post that is contained in the ping, stores this in its database, and sends a status response.
10. If the response is success, Alice's system records the Linkback to prevent re-generation of the request.

Alice's system repeats steps 3, 4, 5, and 10 for each external link found in step 2.

When a visitor requests Bob's post it uses the information stored to mention the Linkback.

## Ping Information

One significant difference between Pingbacks and TrackBacks is the information contained in the ping.

### Pingback

A Pingback ping contains the source document URL and the target document URL; both are required.

### TrackBack

A TrackBack ping contains:

- the source document URL (*required*)
- an excerpt from the source document (*optional*)
- the name of the source blog (*optional*)
- the title of the source document (*optional*)

A reference to the target document is contained in the TrackBack server URL; the format of the TrackBack server URL is implementation specific (see LinkbackAutodiscovery for a description of the Linkback extension implementation).

## License

The Linkback Extension is free software. It is released under the terms of the following BSD License.



Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of PBM Web Development nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## Compatibility

| | Yii |
|---|---|
| **Tested with** | 1.1.7 |
| **Should work with** | 1.1.7 and above |

## Installation

1. Download the Linkback Extension from http://www.yiiframework.com/extension/linkback/
2. Extract the files and place them under the required directory; typically this will be in the extensions directory of the application's blog module.

## Configuration

The Linkback Extension consists of two parts: a Linkback server that accepts Linkback pings from remote clients, and a Linkback client that generates Linkback pings to remote servers.

### Linkback Server

The Linkback server consists of two parts, each separately configured:

- LinkbackAutoDiscovery – generates Pingback and TrackBack server auto-discovery code
- LinkbackController – implements the Pingback and TrackBack servers

#### LinkbackAutoDiscovery

LinkbackAutoDiscovery generates the local Linkback server auto-discovery code and inserts it into posts. It is added as a filter to the controller action that displays posts.

```
public filters() {
   return array(
      //Other filter definitions
      array(
         'path.to.linkback.LinkbackAutoDiscovery +view',
         'url'=>array(
            'model'=>'post',
            'params'=>array(
               'blog'=>'.blog.slug',
               'post'=>'.slug'
            )
         )
      ),
      //Other filter definitions
   );
}
```

Figure 1 - Example LinkbackAutoDiscover Configuration

The configuration above adds the LinkbackAutoDiscover filter to the 'view' action, the model is contained in the controller property 'post', and has URL GET parameters 'blog', which takes the value of the 'slug' attribute in the model's related model 'blog', and 'post' which takes the 'slug' attribute of the model.

### LinkbackController

LinkbackController implements the Pingback and TrackBack servers. It is configured using the CWebApplication::controllerMap property in the application's configuration file (Yii's default configuration file is */protected/config/main.php*):

```
return array(
  // Other CWebApplication configuration
  'controllerMap'=>array(
    'linkback'=>array(
      'class'=>'path.to.LinkbackController',
      'linkback'=>'path.to.linkback.model',
      'params'=>array(
        'urlParam'=>'postModelAttribute'
      ),
      'post'=>'path.to.post.model',
      'rules'=>array(
        'pattern'=>'route'
      ),
      // Other LinkbackController Configuration
      // See LinkbackController::Public Properties
    ),
  ),
  // Other CWebApplication configuration
);
```

Figure 2 - Example LinkbackController Configuration

## Linkback Client

### LinkbackClientBehavior

LinkbackClientBehavior implements the Pingback and TrackBack clients. It is attached to the posts controller using the CController::behaviours() method; the LinkbackClientBehavior::linkback() method is called from the action that displays a post.

> **Note:** The post ***MUST*** be publicly visible, i.e. the post must be approved, published, etc., because the Linkback server may access it to prevent Linkback spamming by ensuring that a) the source URL exists, and b) that the document at the URL (the post) contains the target URL.

```
public function behaviours() {
   return array(
      // Other behavior declarations
      'linkbackClient'=>array(
         'class'=>'path.to.LinkbackClientBehavior',
         // LinkbackClientBehavior properties
      ),
      // Other behavior declarations
   );
}
```

Figure 3 - Example LinkbackClientBehavior Configuration

```
public function actionView() {
   $post = $this->getPost();
   if ($post->isVisible) {
      $this->linkback($post, $this->createAbsoluteUrl('', array(GET
params)));
      $this->render('view', compact('post'));
   }
}
```

Figure 4 - Example LinkbackClientBehavior::linkback() Method Call

## Linkback Server

The Linkback sever consist of two parts; LinkbackAutoDiscovery that add server auto-discovery code to the <head/> element of posts, and LinkbackController that processes Linkback pings from clients.

Two Linkback servers are implemented; a Pingback server and a TrackBack server. The Pingback server is compliant with the Pingback 1.0 specification (http://www.hixie.ch/specs/pingback/pingback#compliant) and the TrackBack with the TrackBack 1.2 specification (http://www.sixapart.com/pronet/docs/trackback_spec).

### LinkbackAutoDiscovery

| Inheritance | LinkbackAutoDiscovery » COutputProcessor » CFilterWidget » CWidget » CBaseController » CComponent |
|---|---|

LinkbackAutoDiscovery adds Pingback and/or TrackBack server auto-discovery code to the <head/> element of posts.

Pingback server auto-discovery code consists of both an X-pingback HTTP header and a <link> element (http://www.hixie.ch/specs/pingback/pingback#TOC2).

TrackBack server auto-discovery code is RDF enclosed in HTML comments in order that the page validates (some validators choke on RDF embedded in XHTML) (http://www.sixapart.com/pronet/docs/trackback_spec). The trackback.ping property is set to the URL of the TrackBack server with the relative URL of the post with respect to the root appended. For example: if the URL to a post is http://example.com/posts/example-post, and the default TrackBack server route is used, the trackback.ping property will be http://example.com/linkback/trackback/posts/example-post.

> **Note:** If a TrackBack server is being implemented, the owner component – the controller – must have a publicly accessible property that contains the post model.

#### Public Properties

See COutputProcessor for inherited properties.

| Name | Type | Description |
|---|---|---|
| **model** | string | The name of the owner property that contains the post model. |
| **servers** | array | Routes to the local Linkback servers. |
| **titleAttribute** | string | The model attribute that contains the document title. |
| **url** | array | URL to the document in the format of key=>value pairs where the keys are the parameter names of CController::createAbsoluteUrl() - route, params, schema, and ampersand - and the values are the parameter values. |

### Public methods

See COutputProcessor for inherited methods.

| Name | Description |
|---|---|
| init() | Initialises the widget. |
| processOutput() | Processes the captured output. |

### Protected methods

| Name | Description |
|---|---|
| addAutoDiscovery() | Adds auto-discovery code for the defined servers. |

### Property Details

#### model

```
public string $model;
```

The name of the owner property that contains the model.

This property is only used in TrackBack auto-discovery code generation and then only if url['params'] references the model or titleAttribute is set.

*Defaults to 'post'*

#### servers

```
public array $servers;
```

Routes to the local Linkback servers.

The array keys are the server types to generate auto-discovery code for; valid keys are LinkbackAutoDiscovery::PINGBACK and LinkbackAutoDiscovery::TRACKBACK.

Array values are URL routes to the servers in the format of ControllerID/ActionID. See CController::createAbsoluteUrl().

*Defaults to array(self::PINGBACK=>'/linkback/pingback',self::TRACKBACK=>'/linkback/trackback')*

#### titleAttribute

```
public string $titleAttribute;
```

The model attribute that contains the document title.

This property is only used in TrackBack auto-discovery code generation and is optional.

*Defaults to 'title'*

#### url

```
public array $url;
```

URL to the document in the format of key=>value pairs where the keys are the parameter names of CController::createAbsoluteUrl() - route, params, schema, and ampersand, and the values are the parameter values; omit keys that take the default values.

The params value is an array of GET parameters in name=>value pairs. If the value of a GET parameter starts with a dot (.) it is an attribute path relative to the model that is resolved at run time; for example array('post'=>'.id') will use the id attribute of the model for the post parameter.

This property is only used in TrackBack auto-discovery code generation.

*Defaults to array();*

## Method Details

### addAutoDiscovery()

| `public string addAutoDiscovery(string $content);` | | |
|---|---|---|
| **$content** | string | The content to which auto-discovery code is added |
| **{return}** | string | The content with auto-discovery code added |

Adds auto-discovery code for the defined servers. The auto-discovery code is placed at the end of the <head/> element.

If the Pingback server is specified the X-Pingback HTTP header and the pingback link element are added.

If the TrackBack server is specified, TrackBack RDF enclosed in HTML comments is added.

### init()

```
public void init();
```

Initialises the widget.

### processOutput()

```
public void processOutput();
```

Processes the captured output.

Adds Linkback server auto-discovery code to the output.

## LinkbackController

| Inheritance | LinkbackController » CExtController » CController » CBaseController » CComponent |
|---|---|

LinkbackController implements the Pingback and TrackBack servers; these accept Linkback pings from clients and store the Linkback information for successful pings in a user defined model.

The Pingback server is compliant with the Pingback 1.0 specification (http://www.hixie.ch/specs/pingback/pingback#compliant).

The TrackBack server is compliant with the TrackBack 1.2 specification (http://www.sixapart.com/pronet/docs/trackback_spec).

A client sends a ping to either the Pingback or TrackBack server (it should not ping both). The pinged server checks that the target URL (which points to the document being linked to by the source document) in the ping exists, that the source URL (which points to the source document that generated the ping) exists and contains the target URL, and that a Linkback between the two documents does not currently exist.

The target URL is parsed in order to check that it points to a valid document; the $params and $rules parameters tell LinkbackController how to do this:

- $rules is an array of URL parsing rules; these are exactly the same us CUrlManager rules; in fact these rules will be the same as those relevant to post URLs in the application configuration.
- $params maps the GET parameters in the URL to attributes of the post model or related models.

For example: If the target URL of a post is http://example.com/posts/post-title, $rules will be something like `array('posts/<pslug:([a-z]+((-|_)[\w]+)*)*>'=>'posts/view')`, which routes the URL to the 'view' action of the 'posts' controller, and puts 'post-title' into `$_GET['pslug']`.

In order that LinkbackController can check the target post exists, the GET parameters must be mapped to post model attributes; so taking the same example $params would look something like `array('pslug'=>'slug')`, which tells LinkbackController that `$_GET['pslug']` maps to the 'slug' attribute of the model defined in $post.

A more complex example is where there is more than one blog; here the URL of a post might be http://example.com/blogs/blog-title/post-title. $rules will be something like array('blogs/<bslug:([a-z]+((-|_)[\w]+)*)*>/<pslug:([a-z]+((-|_)[\w]+)*)*>'=>'blogs/posts/view') which routes the URL to the 'view' action of the 'posts' controller in the 'blogs' module, and puts 'blog-title' into `$_GET['bslug']` and 'post-title' into `$_GET['pslug']`. $params will look something like `array('pslug'=>'slug', 'bslug'=>'blog.slug')`, which maps `$_GET['pslug']`to the 'slug' attribute of the model defined in $post, and `$_GET['bslug']`to the 'slug' attribute of model defined by the 'blog' relationship of the model defined in $post..

If the ping is successful the server creates a record of the Linkback in an application defined model. The server maps internal properties to attributes of the model (see linkbackAttributes). As a minimum the model must store a reference to the document pointed to by the target URL, the

source URL, and the type of ping in order to make the link between them; it may store additional information to provide human readable information about the Linkback that may be available in the ping. The intent is to allow Linkbacks to be stored in the same database table as user submitted comments if required.

> **Note:** If an exception is raised by the server it sent as a response to the client with the error code = 0x810C (Application error). If Yii is in debug mode (YII_DEBUG===TRUE) the exception message is also sent, otherwise the message is set to "Application Error".
>
> The server will raise an exception, for example, if a required property is undefined.

### Public Properties

See CExtController for inherited properties.

| Name | Type | Description |
|---|---|---|
| attributeMap | array | Maps LinkbackController internal properties to linkback model attributes. |
| content | string | Template for content to be stored when a Linkback is registered. |
| contentTypes | array | Content types from which Linkbacks are accepted. |
| encoding | string | Response encoding. |
| excerptEnding | string | The ending added to TrackBack excerpts. |
| excerptExact | boolean | Whether excerptLength is exact. |
| excerptLength | integer | The maximum length of TrackBack excerpts. |
| importPaths | mixed | Paths to import. |
| linkback | string | Path to the model that stores Linkbacks. |
| params | array | Maps target URL parameters to post model attributes. |
| post | string | Path to the model that stores linkbacks. |
| rules | array | Target URL routing rules. |
| scenario | string | The Linkback model scenario when a Linkback is registered. |
| status | mixed | The value given to the linkback model's status attribute. |
| title | string | Template for the title to be saved when a Linkback is registered. |
| validate | boolean | Whether validation is performed when the Linkback is registered. |

## Public methods

See CExtController for inherited methods.

| Name | Description |
|---|---|
| **init()** | Initializes the controller. |
| **actionPingback()** | The Pingback server. |
| **actionTrackback()** | The TrackBack server. |

## Protected methods

See CExtController for inherited methods.

| Name | Description |
|---|---|
| **linkbackRegistered()** | This method is invoked when a Linkback has been registered. |

## Events

| Name | Description |
|---|---|
| **onLinkbackRegistered** | This event is raised when a Linkback has been registered. |

## Property Details

### attributeMap

```
public array $attributeMap;
```

Maps LinkbackController internal properties to linkback model attributes.

Each entry maps an internal property to a linkback model attribute; the key is the LinkbackController property, the value is the linkback model attribute name. Omit any non-required properties that do not have an equivalent attribute in your linkback model.

LinkbackController internal properties are:

- sourceUrl - The source URL *(required)*
- targetId - The id of the target post *(required)*
- type - The type of Linkback; pingback or trackback *(required)*
- blogName - The source blog name (Trackback pingss only)
- content - The content for the Linkback. content defines the template.
- ipAddress - The source IP address
- status - The status of this record on save. Defined by status.
- title - The title of the source post. title defines the template.

*Defaults to array(*
*'sourceUrl' =>'url',*
*'targetId' =>'post_id',*
*'type' =>'type',*
*'blogName' =>'author',*
*'content' =>'content',*
*'ipAddress' =>'ip_address',*
*'status' =>'status',*
*'title' =>'title'*
*);*

### content

```
public string $content;
```

Template for the linkback model's content attribute. The resulting string is saved in the linkback model's content attribute when a Linkback is registered.

See attributeMap for details on how to specify the linkback model's content attribute.

The following placeholders are recognised and will be replaced:

- {excerpt} - An excerpt from the source (TrackBacks only and excerpt must be in the ping).
- {blogName} - The source blog name (TrackBacks only and blog_name must be in the ping).
- {sourceUrl} - The source post URL.
- {title} - Either the title of the source or, if the title is not given, the source URL.
- {type} - The type of Linkback; Pingback or TrackBack.

To make a string containing a placeholder optional (i.e. rendered only if the placeholder has a value) enclose it in square brackets. Escape square brackets that are required in the output with a backslash.

Example: "{type} from {title}[\n\n{excerpt}]" is the equivalent of '{type} from {title}' if excerpt is empty.

*Defaults to "{type} from {title}[\n\n{excerpt}]"*

### contentTypes

```
public array $contentTypes;
```

Content types from which Linkbacks are accepted.

*Defaults to array('text/html', 'text/xml', 'application/xhtml+xml')*

### encoding

```
public string $encoding;
```

Response encoding.

*Defaults to 'utf-8'*

### excerptEnding

```
public string $excerptEnding;
```

The ending added to TrackBack excerpts.

excerptLength includes the length of this property, i.e. the excerpt text will be shortened by the length of this property.

Excerpts are only sent in TrackBack pings.

*Defaults to '…'*

### excerptExact

```
public boolean $excerptExact;
```

Whether the length of TrackBack excerpts is exact.

Only valid if excerptLength is given.

If FALSE the excerpt is truncated on a word boundary such that the excerpt length is less than or equal to excerptLength.

If TRUE the excerpt is truncated to exactly excerptLength.

Excerpts are only sent in TrackBack pings.

*Defaults to FALSE*

### excerptLength

```
public integer $excerptLength;
```

The maximum length of TrackBack excerpts.

*Defaults to 255*

### importPaths

```
public mixed $importPaths;
```

Use to import model inheritances for post and linkback models if required. Leave empty if the models directly extend CActiveRecord or import required inheritances themselves.

An array or comma delimited string of path aliases.

*Defaults to NULL*

### linkback

```
public string $linkback;
```

Path to the model that stores Linkbacks. If the model is imported in importPaths this property need only be the model class name.

This property is required.

*Defaults to NULL*

### params

```
public array $params;
```

Maps target URL parameters to post model attributes.

The format is 'URL param'=>'post model attribute'. Related model attributes are specified using dot notation; e.g. blog.id is the 'id' attribute of the model specified in the 'blog' relationship of the 'post' model.

This property is required.

*Defaults to NULL*

### post

```
public string $post;
```

Path to the model that stores posts. If the model is imported in importPaths this property need only be the model class name.

This property is required.

*Defaults to NULL*

### rules

```
public array $rules;
```

Target URL routing rules (pattern=>route).

This property is required.

*Defaults to NULL*

### scenario

```
public string $scenario;
```

The linkback model scenario when a Linkback is registered.

Use this to ensure the Linkback is not validated against certain linkback model validation rules by setting the value of the rule's 'on' property. For example, if the linkback model is also used for user comments you may have CAPTCHA validation; a Linkback should not be validated against such a rule.

All validation can be turned off by setting validate FALSE.

*Defaults to "linkback"*

### status

```
public mixed $status;
```

The value given to the Linkback's status attribute.

If specified in attributeMap, this value will be stored in the linkback model's attribute defined by attributeMap['status'] and should be set to a value that indicates that the Linkback is approved and/or can be displayed.

*Defaults to "approved"*

### title

```
public string $title;
```

Template for the linkback model's title attribute. The resulting string is stored in the linkback model's title attribute when a Linkback is registered.

See attributeMap for details on how to specify the linkback model's title attribute.

The following placeholders are recognised and will be replaced:

- {blogName} - The source blog name (TrackBacks only and must be in the ping).
- {sourceUrl} - The source post URL.
- {title} - Either the title of the source or the source URL if the title is not given.

To make a string containing a placeholder optional (i.e. rendered only if the placeholder has a value) enclose it in square brackets. Escape square brackets that are required in the output with a backslash.

Example: "{title}[@{blogName}]" is the equivalent of '{title}' if blogName is empty.

*Defaults to "{title}[@{blogName }]"*

### validate

```
public boolean $validate;
```

Whether validation is performed when the Linkback is registered.

*Defaults to TRUE*

## Method Details

### init()

```
public void init()
```

Initialises the controller.

### actionPingback()

```
public void pingback()
```

Implements the Pingback server.

### actionTrackback()

```
public void actionTrackback()
```

Implements the TrackBack server.

### linkbackRegistered()

| protected void linkbackRegistered(CModel $target) | | |
|---|---|---|
| **$target** | CModel | The target post |

This method is invoked when a Linkback has been registered. The default implementation raises the onLinkbackRegistered event.

### onLinkbackRegistered()

| public void onLinkbackRegistered(CEvent $event) | | |
|---|---|---|
| **$event** | CEvent | The event parameter |

This event is raised when a Linkback has been registered.

## Event Details

### linbackRegistered

This event is raised when a Linkback has been registered.

#### Parameters

Parameters are an array of key=>value pairs.

| | | |
|---|---|---|
| **name** | string | The name of the event; 'onLinkbackRegistered' |
| **target** | CModel | The target document |

## Response Codes

Both the Pingback and TrackBack server response codes are those defined in the Pingback 1.0 specification and Specification for Fault Code Interoperability, version 20010516, plus three application defined codes (the Trackback specification only defines the response code for a successful ping; other responses are implementation defined); they are listed below.

| Response Code | | Description | Specification |
|---|---|---|---|
| **0** | **0x0000** | Success - TrackBack only (lack of a code denotes success for a Pingback) | TrackBack |
| **16** | **0x0010** | Source URI does not exist. | Pingback |
| **17** | **0x0011** | Source URI does not link to target URI. | Pingback |
| **18** | **0x0012** | Source URI not specified. | Application |
| **32** | **0x0020** | Target URI does not exist. | Pingback |
| **33** | **0x0021** | Target URI cannot be used. | Pingback |
| **48** | **0x0030** | Linkback already registered. | Pingback |
| **49** | **0x0031** | Access denied. | Pingback |
| **50** | **0x0032** | Linkback not registered. | Application |
| **51** | **0x0033** | Invalid content type. | Application |
| **-32700** | **0x8044** | xmlrpc parse error;  not well formed. | SFCI |
| **-32600** | **0x80A8** | Server error.  Invalid xml-rpc; not conforming to spec. | SFCI |
| **-32601** | **0x80A7** | Server error.  Requested method not found. | SFCI |
| **-32602** | **0x80A6** | Server error.  Invalid method parameters. | SFCI |
| **-32500** | **0x810C** | Application error (details of the error are in the message) | SFCI |

**Note:** Server errors are logged.

**WARNING: If you have debug turned on that outputs to the browser, server responses may be seen by clients as non-well formed XML due to the extra content.**

## Linkback Client

### LinkbackClientBehavior

| Inheritance | LinkbackClientBehavior » CBehavior » CComponent |
|---|---|

LinkbackClientBehavior implements Pingback and TrackBack clients.

The Pingback client is compliant with the Pingback 1.0 specification (http://www.hixie.ch/specs/pingback/pingback#compliant).

The TrackBack client is compliant with the TrackBack 1.2 specification (http://www.sixapart.com/pronet/docs/trackback_spec).

Posts are parsed to extract external links which are then searched for Pingback or TrackBack server auto-discovery code; if a server is found a Linkback ping is sent to it. If the Linkback ping is successful it is recorded to prevent further pings being sent to the server about the post. The Linkback is also registered if the sever response code to a Pingback ping is 0x0030 (Linkback already registered).

LinkbackClientBehavior uses two database tables that are automatically created if they do not exist; these record successful Linkbacks to prevent further pings for them, and which posts are currently pinging to prevent duplicate pings if the server accesses the post when pinged. The schema for these is are given in Appendix A – Linkback Client Database Table Schema.

LinkbackClientBehavior sends server responses to the Yii::trace() method to assist debugging.

#### Public Properties

See CBehavior for inherited properties.

| Name | Type | Description |
|---|---|---|
| **attributes** | mixed | The post attributes parsed for external links. |
| **blogName** | string | The name of the blog. |
| **encoding** | string | Pingback ping encoding. |
| **excerptAttribute** | string | The post attribute used for TrackBack excerpts. |
| **excerptDelimiter** | string | The string used to truncate TrackBack excerpts. |
| **excerptEnding** | string | The ending added to TrackBack excerpts. |
| **excerptExact** | boolean | Whether excerptLength is exact. |
| **excerptLength** | integer | The maximum length of TrackBack excerpts. |
| **linkbacks** | mixed | The types of Linkbacks we can send. |
| **linkbacksTable** | string | The name of the table used to record successful Linkbacks. |

# Linkback Extension

| Name | Type | Description |
|---|---|---|
| pingsTable | string | The name of the table used to record current pings. |
| titleAttribute | string | The post title attribute. |

## Public methods

See CBehavior for inherited methods.

| Name | Description |
|---|---|
| attach() | Attaches the behavior object to the component. |
| linkback() | Generate pings. |

## Protected methods

See CBehavior for inherited methods.

| Name | Description |
|---|---|
| linkbackRegistered() | This method is invoked when a Linkback has been registered. |

## Events

| Name | Description |
|---|---|
| onLinkbackRegistered | This event is raised when a Linkback has been registered. |

## Property Details

### attributes

**public mixed $attributes;**

The post attributes parsed for external links.

Either an array of attribute names or a string that is a comma delimited list of attribute names.

URLs found in these attributes are searched for Linkback server auto-detect code and a ping sent to the server if found.

The content of these attributes must be visible in the post when it is read.

*Defaults to 'content'*

### blogName

**public string $blogName**;

The name of the blog.

Set empty to not send the blog name.

The blog name can also be set "on the fly" when generating Linkbacks for a given post (see linkback()).

The blog name is only sent in TrackBack pings.

*Defaults to NULL*

### encoding

**public string encoding;**

Pingback ping encoding.

*Defaults to 'utf-8'*

### excerptAttribute

**public string $excerptAttribute;**

The post attribute used for TrackBack excerpts.

Excerpts are only sent in TrackBack pings.

*Defaults to 'content'*

### excerptDelimiter

**public string $excerptDelimiter;**

The string used to truncate TrackBack excerpts.

The excerpt will be up to and including the first occurrence of the delimiter; this makes it easy to – for example – use the first sentence as the excerpt. The excerpt may be further truncated to excerptLength.

Set empty not to truncate excerpts using a delimiter.

Excerpts are only sent in TrackBack pings.

*Defaults to '.'*

### excerptEnding

**public string $excerptEnding;**

The ending added to TrackBack excerpts.

excerptLength includes the length of this property, i.e. the excerpt text will be shortened by the length of this property.

Excerpts are only sent in TrackBack pings.

*Defaults to '...'*

### excerptExact

**public boolean $excerptExact;**

Whether the length of TrackBack excerpts is exact.

Only valid if excerptLength is given.

If FALSE the excerpt is truncated on a word boundary such that the excerpt length is less than or equal to excerptLength.

If TRUE the excerpt is truncated to exactly excerptLength.

Excerpts are only sent in TrackBack pings.

*Defaults to FALSE*

### excerptLength

**public integer $excerptLength;**

The maximum length of TrackBack excerpts.

Set empty not to truncate excerpts.

Excerpts are only sent in TrackBack pings.

*Defaults to 255*

### linkbacks

**public mixed $linkbacks;**

The types of Linkbacks we can send.

Either an array or a comma delimited string of Linkback types. Either an array or a comma delimited string of Linkback types. Servers are searched for in the order given; the first discovered server is used.

Valid Linkback types are 'pingback' and 'trackback'.

*Defaults to 'pingback, trackback'*

### linkbacksTable

**public string $linkbacksTable;**

The name of the table used to record successful Linkbacks.

*Defaults to 'linkbacks'*

### pingsTable

**public string $pingsTable;**

The name of the table used to record current pings.

*Defaults to 'linkback_pings'*

### titleAttribute

| public string $titleAttribute; |
|---|

The post title attribute.

Set empty to not send the title.

The post title is only sent in TrackBack pings.

*Defaults to 'title'*

## Method Details

### attach()

| public void attach(CComponent $owner) |
|---|

Attaches the behavior object to the component. Sets the owner property, attaches event handlers as declared in events, and creates the linkbackTable and pingsTable if they do not exist. Make sure you call the parent implementation if you override this method.

### linkback()

| public void linkback(CModel $post, string $sourceUrl , $blogName=null); | | |
|---|---|---|
| **$post** | CModel | The post |
| **$sourceUrl** | string | The absolute URL to the post |
| **$blogName** | string | The name of the blog. If set, this overwrites the blog name set in the configuration. Used in multi-blog applications where the blog is determined from the post. |

Generates Linkback pings to external URLs in the post that have a Linkback server.

### linkbackRegistered()

| protected void linkbackRegistered(CModel $post, string $targetUrl) | | |
|---|---|---|
| **$post** | CModel | The post that generated the linkback |
| **$targetUrl** | string | The target URL of the ping |

This method is invoked when a Linkback has been registered. The default implementation raises the onLinkbackRegistered event.

### onLinkbackRegistered()

| public void onLinkbackRegistered(CEvent $event) | | |
|---|---|---|
| **$event** | CEvent | The event parameter |

This event is raised when a Linkback has been registered.

## Event Details

### onLinbackRegistered

This event is raised when a Linkback has been registered.

### Parameters

Parameters are an array of key=>value pairs.

| | | |
|---|---|---|
| **name** | string | The name of the event; 'onLinkbackRegistered'. |
| **post** | CModel | The source post. |
| **targetUrl** | String | The target URL. |

## Appendix A – Linkback Client Database Table Schema

### Linkbacks Table

```
CREATE TABLE `linkbacks` (
    `post_id` int(11) UNSIGNED NOT NULL ,
    `target_url` varchar(255) NOT NULL ,
    PRIMARY KEY (`post_id`, `target_url`)
);
```

### LinkbackPings Table

```
CREATE TABLE `linkback_pings` (
    `post_id` int(11) UNSIGNED NOT NULL ,
);
```