

This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. The article is believed to be still valid for the original topic scope.



## Contents

- [1 The service](#)
  - ◆ [1.1 Scaling](#)
  - ◆ [1.2 Service actions](#)
    - ◇ [1.2.1 Action geturl](#)
      - [1.2.1.1 Arguments](#)
      - [1.2.1.2 Errors](#)
    - ◇ [1.2.2 Action getres](#)
      - [1.2.2.1 Errors](#)
    - ◇ [1.2.3 Action getwidgeticon](#)
      - [1.2.3.1 Errors](#)
    - ◇ [1.2.4 Action getavatar](#)
      - [1.2.4.1 Errors](#)
- [2 Code example](#)
- [3 See also](#)

## The service

The image service delivers all images to the client scripts (widgets) in a format that is supported in the mobile terminal.

In the usual case, images are converted to 8 bit indexed color PNG format with the colors reduced. You can also request 24 bit PNG or JPEG images. However, note that only 8 bit PNGs are guaranteed to work on all MIDP 2.0 mobile Java implementations.

**Note.** The old picviewer service has been removed and its functionality has been separated into Image service and [Syndication service](#). Now the sole purpose of Image service is to fetch images on demand. Syndication service takes care of delivering feed content such as Flickr image feeds.

## Image\_service

The service consists of four actions of which only the first is necessary in normal development:

Action	Generic	Function
<b><u>geturl</u></b>	yes	Receive an image scaled to the size specified in the call. The caller can set a bounding box into which the image is fitted. Other optional arguments are also possible.
<b><u>getres</u></b>	yes	Gets a widget image resource with the id.
<b><u>getwidgeticon</u></b>	yes	Gets a widget icon using the widget id.
<b><u>getavatar</u></b>	yes	Gets an user avatar using the user name.

## Scaling

Images are scaled using a bounding box. The image is resized so that it fits completely into the dimensions of the box. The original aspect ratio of the image is conserved. Maximum dimensions for the box are 1920x1920 pixels. Also, source images larger than this are scaled down to fit into the maximum box if neither bounding box arguments are given.

## Service actions

The following sections describe the specifications for the available service actions.

### Action `geturl`

```
namespace geturl is ContentAction
{
  input = (list (bag (bind (const url) (string url))
                    (optional (bind (const boundx) (int boundx)))
                    (optional (bind (const boundy) (int boundy)))
                    (optional (bind (const minsize) (int bytesize)))
                    (optional (bind (const minwidth) (int minwidth)))
                    (optional (bind (const minheight) (int minheight)))
                    (optional (bind (const refreshtime) (int seconds)))
                    (optional (bind (const format)
                                   (choice (const png8)
                                           (const png24)
                                           (const jpeg)))))))

  output = (bytes imageData)

  generic = true
}
```

## Image\_service

### Arguments

The only required argument is *url*. The rest are optional.

name	type	Description
<b>url</b>	<b>string</b>	<b>The URL of the image.</b>
boundx	int	The width of the bounding box.
boundy	int	The height of the bounding box.
minsize	int	The minimum size of the image in bytes of the source image.
minwidth	int	The minimum width of the image in pixels of the source image.
minheight	int	The minimum height of the image in pixels of the source image.
refreshtime	int	Caching time of the image in seconds. If this argument is not given the default will be 10800s. The minimum is 60s.
format	png8   png24   jpeg	The desired encode format. Please use png-images for maximum compatibility.

### Errors

Code	Message	Description
error	Unable to get image: <message>	Something went wrong. For example, the server returned 404 for the requested URL.

### Action `get:res`

```
namespace get:res is ContentAction
{
  input = (list (bag (bind (const id) (int resourceId))
    (optional (bind (const boundx) (int boundx)))
    (optional (bind (const boundy) (int boundy)))
    (optional (bind (const format)
      (choice (const png8)
        (const png24)
        (const jpeg)))))))

  output = (bytes resourceData)

  generic = true
}
```

### Errors

Code	Message	Description
------	---------	-------------

## Image\_service

notfound	Image not found or access denied.	No such resource id, or user trying to get resource of a PRIVATE widget belonging to another user.
notimage	The requested resource was not an image.	The requested resource was of some other type.
error	Unable to get image: <message>	Something went wrong. For example, a temporary error.

### Action `getwidgeticon`

```
namespace getwidgeticon is ContentAction
{
  input = (int widgetId)

  output = (bytes iconResourceData)

  generic = true
}
```

### Errors

Code	Message	Description
notfound	Image not found or access denied.	No such icon
error	Unable to get image: <message>	Something went wrong, for example, a temporary error.

### Action `getavatar`

```
namespace getavatar is ContentAction
{
  input = (list (bag (bind (const username) (string username))
                  (optional (bind (const boundx) (int boundx)))
                  (optional (bind (const boundy) (int boundy)))
                  (optional (bind (const format)
                                (choice (const png8)
                                       (const png24)
                                       (const jpeg)))))))

  output = (bytes avatarImageData)

  generic = true
}
```

### Errors

Code	Message	Description
------	---------	-------------

## Image\_service

notfound	Image not found or access denied.	No such avatar.
error	Unable to get image: <message>.	Something went wrong. For example, a temporary error.

## Code example

[Example Image Service](#)

## See also

- [Getting content with Services](#)
- [Available content fetching services](#)
  - ◆ [Syndication service](#)
  - ◆ [Webfeed service](#)
  - ◆ [HTTP service](#)
- [Fetcher details](#)
  - ◆ [Feed formats](#)
  - ◆ [HTTP authentication](#)
- [Advanced filters](#)
  - ◆ [Filter expressions reference](#)