

STREAMEZZO RICH MEDIA SERVER

Remote Cache Management

This document is the property of Streamezzo. It cannot be distributed without the authorization of Streamezzo.

Table of contents

STREAMEZZO RICH MEDIA SERVER.....	1
Remote Cache Management.....	1
Table of contents.....	3
<u>1. INTRODUCTION.....</u>	<u>4</u>
<u>2. OVERVIEW.....</u>	<u>5</u>
<u>3. RICH MEDIA SERVER REMOTE CACHE MANAGEMENT LIBRARY.....</u>	<u>6</u>
<u>3.1 Installation.....</u>	<u>6</u>
<u>3.2 Remote cache manager configuration.....</u>	<u>7</u>
<u>3.3 HTTP/XML API based remote cache management.....</u>	<u>9</u>
<u>3.4 Web service based remote cache management.....</u>	<u>10</u>
<u>3.5 Common cache management operations.....</u>	<u>11</u>
<u>3.5.1. Removing all expired cached scenes.....</u>	<u>11</u>
<u>3.5.2. Removing all expired cached resources.....</u>	<u>12</u>
<u>3.5.3. Removing all cache scenes.....</u>	<u>12</u>
<u>3.5.4. Removing all cached resources.....</u>	<u>13</u>
<u>3.6 HTTP/XML API only remote cache management operations.....</u>	<u>15</u>
<u>3.6.1. Removing a single cached resource.....</u>	<u>15</u>
<u>3.6.2. Prefetching of a cache entry for a given scene and a user-agent</u>	<u>16</u>
.....	

1. INTRODUCTION

Streamezzo Rich Media software suite provides many ways for caching data so that end user experience is greatly improved since latency is reduced when navigating through the service. Rich Media Client (RMC) offers the cached objects mechanism to store some scenes onto the user devices. Rich Media Server offers 2 levels of data caching. First cache level is used to store complete generated scenes to be shared among several service end users which devices share the same profile. Second cache level is supposed to store data used as input resources for building service scenes. They can be either media resources such as images, audio and video contents, or a service managed persistent resource such as a subscription registry.

Rich Media Server cache contents can be managed through its administrative web interface (see *Rich Media Server 4.0 Administration Guide* for more information). Rich Media Server 4.0.9.2 introduces two ways to manage it remotely: HTTP/XML API or web service.

If remote management is supposed to be integrated into some Java applications, Rich Media Server release provides a Java client for both API.

This document aims at describing how to remotely manage Rich Media Server cache by each one of these ways.

2. OVERVIEW

Since Rich Media Server 4.0.9.2, Rich media cached scenes (level 1) as well as cached resources (level 2) can be remotely managed by any external programs (Content Management Systems for example). As an administration feature, it relies on the same authentication mechanism as administrative web console and Studio publication process. You are supposed to define the login and password couple used to authenticating user. Then you have access to cached contents according to your privileges. As a Rich Media Server administrator, you can manage remotely the whole cache content whereas you have only access to content related to the services you are allowed to manage if you are only declared as an author.

Available methods provide a way to:

- remove all expired cached scenes
- remove all expired cached resources
- delete all cached scene
- delete all cached resources
- delete a specific cached resource

These remote services are called through HTTP using either HTTP/XML API or web service.

There are some more methods available for HTTP/XML API only for:

- prefetching a scene to cache on Rich Media Server Front directly
- prefetching a scene to cache on Rich Media Server Front through Administration

Prefetching is made possible through Rich Media Server Administration because Rich Media Server Front may not be reachable directly. Indeed, it may be isolated in a front end which is not supposed to be reachable from back end for security purpose.

In order to integrate remote cache management quickly, a Java client is provided within Rich Media Server release CDROM. It gives access to an API so that a Java CMS could easily include these features. Using this library is not mandatory but strongly recommended. Next parts of this document explain in more details how to manage Rich Media Server cache remotely with some source code examples and figures.

3. RICH MEDIA SERVER REMOTE CACHE MANAGEMENT LIBRARY

Rich Media Server release CDROM includes the file *Distant Manager/stzManager.tar.gz*.

3.1 Installation

Uncompress the file:

- base directory includes remote cache manager library *stzManager.jar* as well as every required third party libraries
- *javadoc* directory contains java documentation of the API

You are supposed to add *stzManager.jar* into **class path** as well as every provided third party java libraries.

For using remote cache management client through any API, following libraries are supposed to be added into class path:

- *catalinaBase64.jar*
- *log4j-1.2.7.jar*
- *stzServerLogger.jar*

For using HTTP/XML API, following libraries are supposed to be added into class path:

- *commons-httpclient-2.0.jar*
- *commons-logging-api.jar*
- *stzHTTPClient.jar*

For using Web Service, following libraries are supposed to be added into class path:

- *axis.jar*
- *commons-discovery-0.2.jar*
- *jaxrpc.jar*
- *saaj.jar*
- *wsdl4j-1.5.1.jar*

Every library is included into the remote manager deliverable (*stzManager.tar.gz*).

You should then be able to use the remote cache manager API from your java project.

3.2 Remote cache manager configuration

Remote cache manager requires some parameters to be set in order to contact Rich Media Server Administration. For example, you are supposed to define login and password for Rich Media Server authentication. According to your privileges you will be able to clean or clear only scenes and resources you are authorized to manage (see *Rich Media Server Administration Guide* for more information).

These parameters are supposed to be set into a separate file named *stzManager.properties*, located into a directory included the application class path. A default configuration file is provided within *stzManager.tar.gz* in order to help you defining required parameters.

Following parameters must be set according to your environment:

- Host name or IP address of the Rich Media Server admin server:

```
com.streamezzo.servicenode.host=<hostname>
```

⇒ Replace *<hostname>* with the Rich Media Server host name or IP address.

- Port number of the Rich Media Server admin server:

```
com.streamezzo.servicenode.port=<port_number>
```

⇒ Replace *<port_number>* with the Rich Media Server port number.

- The login of your Rich Media Server admin account:

```
com.streamezzo.user.login=<login>
```

⇒ Replace *<login>* with a Rich Media Server account login.

- The password of your Rich Media Server admin account:

```
com.streamezzo.user.password=<password>
```

⇒ Replace *<password>* with a Rich Media Server account password.

- Enable or disable the use of proxy:

```
com.streamezzo.proxy.set=[true|false]
```

⇒ Set true if you use a proxy server.

- Host name or IP address of your proxy server:

```
com.streamezzo.proxy.host=<hostname>
```

⇒ If property *com.streamezzo.proxy.set* is true then replace *<hostname>* with the host name or the IP address of your proxy server.

- Port number of your proxy server:

```
com.streamezzo.proxy.port=<port_number>
```

⇒ If the property *com.streamezzo.proxy.set* is true then replace *<port_number>* with the port number of your proxy server.

For example, you can see below a sample configuration. For your information, this is the default configuration:

```
com.streamezzo.servicenode.host=www.streamezzo.com  
com.streamezzo.servicenode.port=8080  
com.streamezzo.user.login=stzadmin  
com.streamezzo.user.password=stzadmin
```

3.3 HTTP/XML API based remote cache management

HTTP/XML Remote Cache Management API relies on HTTP requests returning XML message as response. Requests are supposed to target Rich Media Server Administration using the header named *User-Agent* set to *Streamezzo/StzServerAdminClient* and including the parameter named *step* to define the operation to achieve.

This remote cache manager designed to use HTTP requests to interact with the Rich Media Server.

The easier way to create an instance of *RemoteCacheManager*, is to use the *RemoteManagerFactory*. It provides two methods to retrieve an instance of *RemoteCacheManager*.

Here are below two samples introducing the two ways to retrieve an instance of *RemoteCacheManager*.

Example 1: Using *RemoteManagerFactory* to create an instance of *RemoteCacheManager* with a specific configuration stored in the file *configuration.properties*.

```
public class RemoteCacheManagerClient
{
    private InputStream configProps;
    private RemoteCacheManager manager;

    protected void init() throws Exception
    {
        configProps=this.getClass().getResourceAsStream("config.properties"
        );
        manager=RemoteManagerFactory.getRemoteCacheManager(configProps);
    }
}
```

Example 2: Using *RemoteManagerFactory* to create an instance of *RemoteCacheManager* with the default configuration.

```
public class RemoteCacheManagerClient
{
    private RemoteCacheManager manager;

    protected void init() throws Exception
    {
        manager = RemoteManagerFactory.getRemoteCacheManager();
    }
}
```

3.4 Web service based remote cache management

There is a way to manage Rich Media Server cache remotely using a web service implemented by Rich Media Server Administration. Rich Media Server remote cache management web service is based upon Axis 1.4 framework. Web service is fully described by WSDL to download from:

<http://host:port/stzadmin/services/RemoteCacheManager?WSDL>

where host and port match Rich Media Server Administration server IP host and port number.

StzManager library API makes easy to use it from a Java application by using the java class *com.streamezzo.manager.webservice.RemoteCacheManager*. To instantiate it you are supposed to use the *RemoteManagerFactory*. It provides two methods to retrieve an instance of *RemoteCacheManager*.

Here are two samples introducing the two ways to retrieve an instance of *RemoteCacheManager*.

Example 1: Using *RemoteManagerFactory* to create an instance of *RemoteCacheManager* with a specific configuration stored in the file *configuration.properties*.

```
public class RemoteCacheManagerClient
{
    private InputStream configProps;
    private RemoteCacheManager manager;

    protected void init()
    {
        configProps=this.getClass().getResourceAsStream("config.properties");
        manager=RemoteManagerFactory.getRemoteCacheManagerWebService(configProps);
    }
}
```

Example 2: Using *RemoteManagerFactory* to create an instance of *RemoteCacheManager* with the default configuration.

```
public class RemoteCacheManagerClient
{
    private RemoteCacheManager manager;

    protected void init ()
    {
        manager=RemoteManagerFactory.getRemoteCacheManagerWebService();
    }
}
```

3.5 Common cache management operations

Every operation excepted scenes prefetching is available through either HTTP/XML API, web service or stzManager library API. Each operation is described in details in the following chapters.

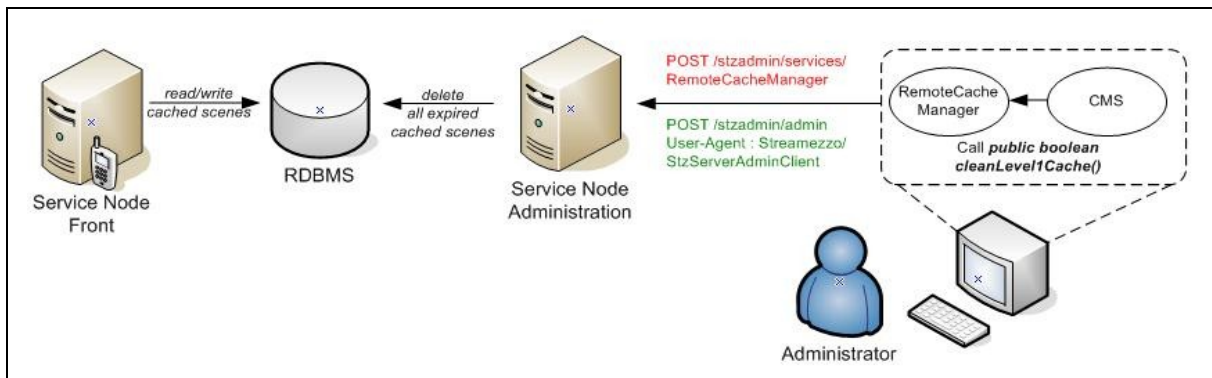
3.5.1. Removing all expired cached scenes

This method allows removing all expired scene cache entries (level 1).

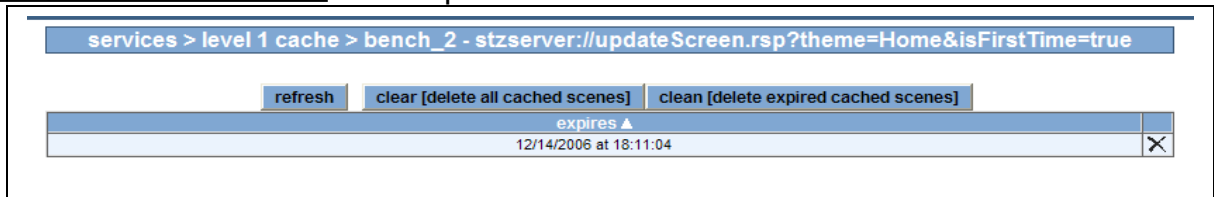
HTTP/XML API step parameter value equals 57. Successful response is supposed to be:

```
<Admin>
  <ok>true</ok>
  <return>true</return>
  <message>Cache L1 has been cleaned</message>
</Admin>
```

The execution process is illustrated below:



12/14/2006 at 6:55 PM: one expired cached scene exists in level 1 cache



The 12/14/2006 at 7:00 PM: stzManager API used to remove all expired cache entries



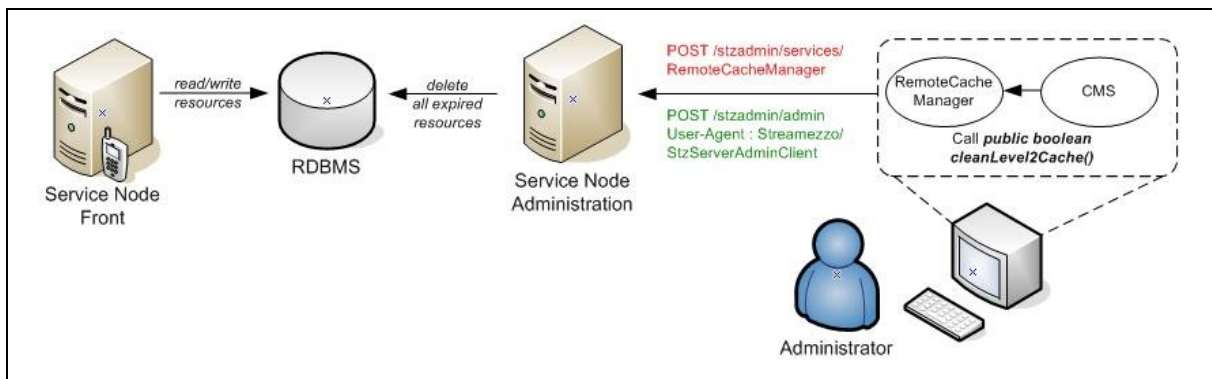
3.5.2. Removing all expired cached resources

This method allows removing all expired external resources cache entries (level 2).

HTTP/XML API step parameter value equals 54. Successful response is supposed to be:

```
<Admin>
  <ok>true</ok>
  <return>true</return>
  <message>Cache L2 has been cleaned</message>
</Admin>
```

The execution process is illustrated below:



12/15/2006 at 9:05 AM: some expired cached resources exist

services > level 2 cache				
<input type="button" value="refresh"/> <input type="button" value="clear [delete all resources]"/> <input type="button" value="clean [delete expired resources]"/>				
service ▲	author	resource key	expires	
bench_2	elgringo	http://demo.streamezzo.com:80/bench/videos/p-3084.mpg-c1.3gp	12/15/2006 at 05:49:33	✗
bench_2	elgringo	http://demo.streamezzo.com:80/bench/videos/p-10105.mpg-c1.3gp	12/15/2006 at 05:50:15	✗
bench_2	elgringo	http://demo.streamezzo.com:80/bench/videos/p-8791.mpg-c1.3gp	12/15/2006 at 05:50:18	✗
bench_2	elgringo	http://demo.streamezzo.com:80/bench/videos/p-12261.mpg-c1.3gp	12/15/2006 at 05:50:23	✗

12/15/2006 at 9:15 AM: stzManager API used to remove all expired resources

services > level 2 cache	
<input type="button" value="refresh"/>	
Cache is empty: no cached resources available.	

3.5.3. Removing all cache scenes

This method allows deleting all cached scenes (level 1), expired as well as valid ones.

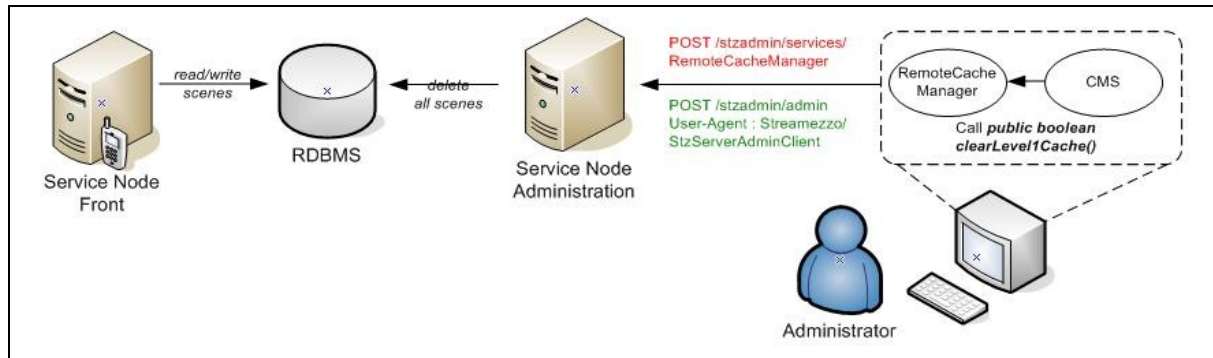
HTTP/XML API step parameter value equals 56. Successful response is supposed to be:

```

<Admin>
  <ok>true</ok>
  <return>true</return>
  <message>Cache L1 has been cleared</message>
</Admin>

```

The execution process is illustrated below:



12/14/2006 at 6:02 PM: some cached scenes exist

services > level 1 cache				
<input type="button" value="refresh"/> <input type="button" value="clear [delete all cached scenes]"/> <input type="button" value="clean [delete expired cached scenes]"/>				
service ▲	server	URL	occurrences	
bench_2	localhost	stzserver://updateScreen.rsp?theme=Home&isFirstTime=true	1	✗
bench_2	localhost	stzserver://launchVideo.rsp?video=http://demo.streamezzo.com/bench/videos/P-8791.mpg-c1.3gp&theme=Home&rubrique=No&index=4	1	✗
bench_2	localhost	stzserver://launchVideo.rsp?video=http://demo.streamezzo.com/bench/videos/P-3084.mpg-c1.3gp&theme=Home&rubrique=No&index=1	1	✗
bench_2	localhost	stzserver://launchVideo.rsp?video=http://demo.streamezzo.com/bench/videos/P-12261.mpg-c1.3gp&theme=Home&rubrique=No&index=0	1	✗
bench_2	localhost	stzserver://launchVideo.rsp?video=http://demo.streamezzo.com/bench/videos/P-10105.mpg-c1.3gp&theme=Home&rubrique=No&index=3	1	✗

12/14/2006 at 6:15 PM: stzManager API used to remove all cached scenes

services > level 1 cache	
<input type="button" value="refresh"/>	
Cache is empty: no cached scenes available.	

3.5.4. Removing all cached resources

This method allows removing all cached resources (level 2), expired as well as valid ones.

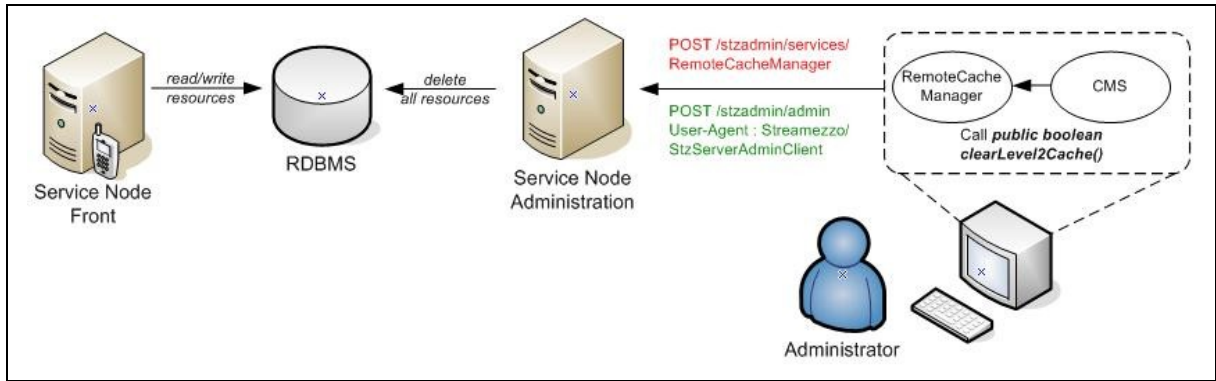
HTTP/XML API step parameter value equals 53. Successful response is supposed to be:

```

<Admin>
  <ok>true</ok>
  <return>true</return>
  <message>Cache L2 has been cleared</message>
</Admin>

```

The execution process is illustrated below:



12/14/2006 at 6:02 PM: some cached resources exist

services > level 2 cache				
refresh				
clear [delete all resources]				
clean [delete expired resources]				
service ▲	author	resource key	expires	
bench_2	elgringo	http://demo.streamezzo.com:80/bench/videos/p-3084.mpg-c1.3gp	12/15/2006 at 05:49:33	✕
bench_2	elgringo	http://demo.streamezzo.com:80/bench/videos/p-10105.mpg-c1.3gp	12/15/2006 at 05:50:15	✕
bench_2	elgringo	http://demo.streamezzo.com:80/bench/videos/p-8791.mpg-c1.3gp	12/15/2006 at 05:50:18	✕
bench_2	elgringo	http://demo.streamezzo.com:80/bench/videos/p-12261.mpg-c1.3gp	12/15/2006 at 05:50:23	✕

12/14/2006 at 6:15 PM: stzManager API used to remove all cached resources

services > level 2 cache
refresh
Cache is empty: no cached resources available.

3.6 HTTP/XML API only remote cache management operations

Following operations are not available through Rich Media Server Remote Cache Manager web service. They can only be called through HTTP/XML API or StzManager library API.

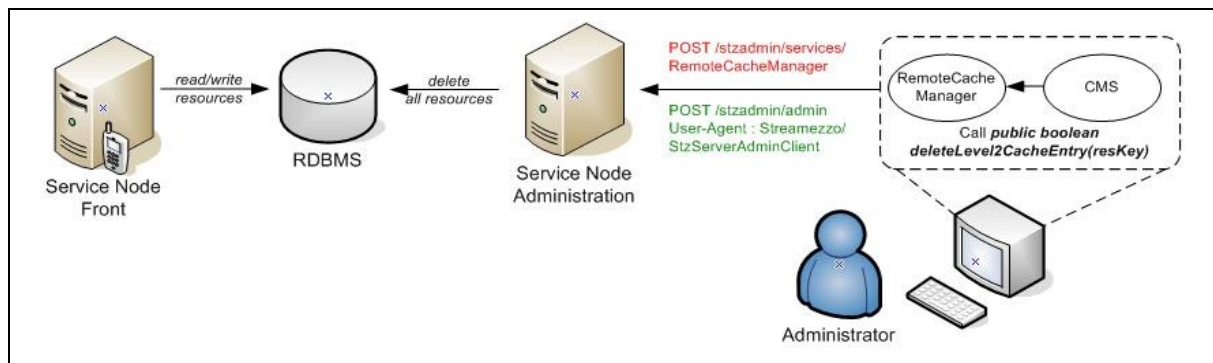
3.6.1. Removing a single cached resource

This method allows deleting a single cached resource (level 2).

HTTP/XML API step parameter value equals 55. Successful response is supposed to be:

```
<Admin>
  <ok>true</ok>
  <return>true</return>
  <message>The resource has been deleted</message>
</Admin>
```

The execution process is illustrated below:



The method is named *deleteLevel2CacheEntry*. It takes the resource key as parameter. Resource key is supposed to be either resource URL or resource path with extension “*_serviceID*” where *serviceID* stands for the service identifier assigned by Rich Media Server at service first publication time. Resource key appears in level 2 cache management page of Rich Media Server Administration web interface.

services > level 2 cache				
<input type="button" value="refresh"/> <input type="button" value="clear [delete all resources]"/> <input type="button" value="clean [delete expired resources]"/>				
service ▲	author	resource key	expires	
bench_2	elgringo	http://demo.streamezzo.com:80/bench/vignettes/irobot.196x89.png	12/14/2006 at 18:06:31	✕
bench_2	elgringo	http://demo.streamezzo.com:80/bench/vignettes/air.94x47.png	12/14/2006 at 18:06:31	✕
bench_2	elgringo	http://demo.streamezzo.com:80/bench/vignettes/schrek.94x47.png	12/14/2006 at 18:06:31	✕
bench_2	elgringo	http://demo.streamezzo.com:80/bench/vignettes/sexypinup.94x47.png	12/14/2006 at 18:06:31	✕
bench_2	elgringo	http://demo.streamezzo.com:80/bench/vignettes/zidane.94x47.png	12/14/2006 at 18:06:31	✕
bench_2	elgringo	http://demo.streamezzo.com:80/bench/videos/p-3084.mpg-c1.3gp	12/15/2006 at 05:49:33	✕
bench_2	elgringo	http://demo.streamezzo.com:80/bench/videos/p-10105.mpg-c1.3gp	12/15/2006 at 05:50:15	✕
bench_2	elgringo	http://demo.streamezzo.com:80/bench/videos/p-8791.mpg-c1.3gp	12/15/2006 at 05:50:18	✕
bench_2	elgringo	http://demo.streamezzo.com:80/bench/videos/p-12261.mpg-c1.3gp	12/15/2006 at 05:50:23	✕

After calling:

deleteLevel2CacheEntry("http://demo.streamezzo.com:80/bench/vignettes/schrek94x47.png")

the external resource has disappeared from the list.

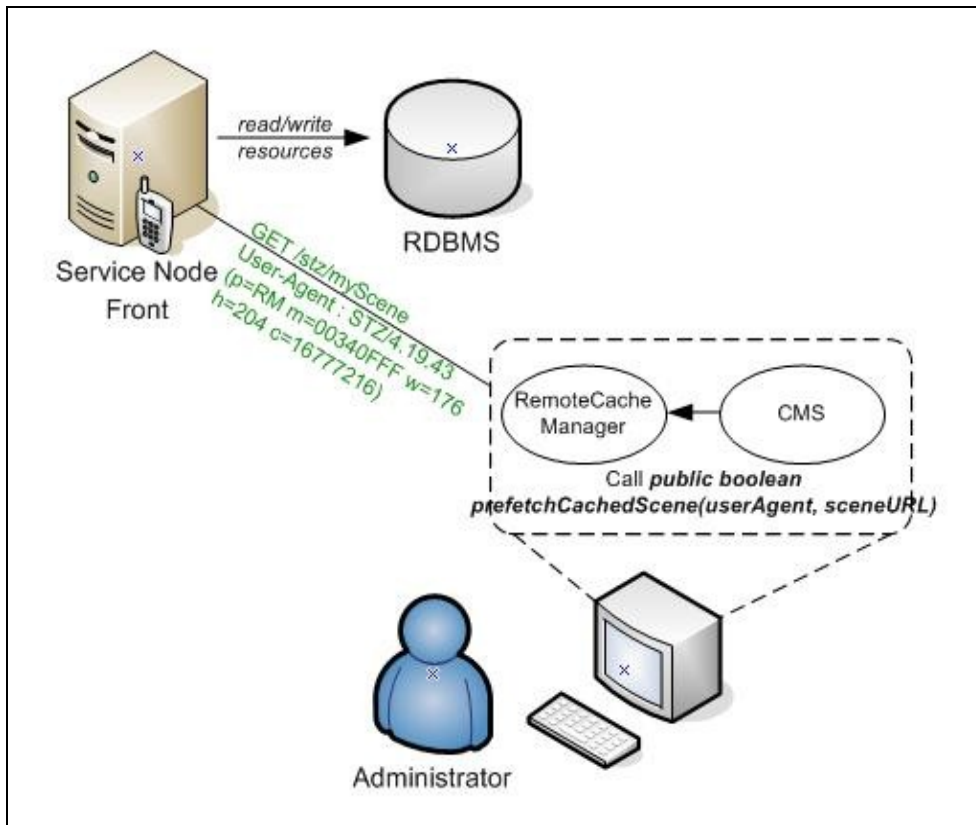
services > level 2 cache				
<input type="button" value="refresh"/> <input type="button" value="clear [delete all resources]"/> <input type="button" value="clean [delete expired resources]"/>				
service ▲	author	resource key	expires	
bench_2	elgringo	http://demo.streamezzo.com:80/bench/vignettes/irobot.196x89.png	12/14/2006 at 18:06:31	✕
bench_2	elgringo	http://demo.streamezzo.com:80/bench/vignettes/air.94x47.png	12/14/2006 at 18:06:31	✕
bench_2	elgringo	http://demo.streamezzo.com:80/bench/vignettes/sexypinup.94x47.png	12/14/2006 at 18:06:31	✕
bench_2	elgringo	http://demo.streamezzo.com:80/bench/vignettes/zidane.94x47.png	12/14/2006 at 18:06:31	✕
bench_2	elgringo	http://demo.streamezzo.com:80/bench/videos/p-3084.mpg-c1.3gp	12/15/2006 at 05:49:33	✕
bench_2	elgringo	http://demo.streamezzo.com:80/bench/videos/p-10105.mpg-c1.3gp	12/15/2006 at 05:50:15	✕
bench_2	elgringo	http://demo.streamezzo.com:80/bench/videos/p-8791.mpg-c1.3gp	12/15/2006 at 05:50:18	✕
bench_2	elgringo	http://demo.streamezzo.com:80/bench/videos/p-12261.mpg-c1.3gp	12/15/2006 at 05:50:23	✕

3.6.2. Prefetching of a cache entry for a given scene and a user-agent

Prefetching a cached scene consists in simulating a end user request towards a service scene which is cache enabled. Thus the scene is registered into level 1 cache, without any end user request towards that scene.

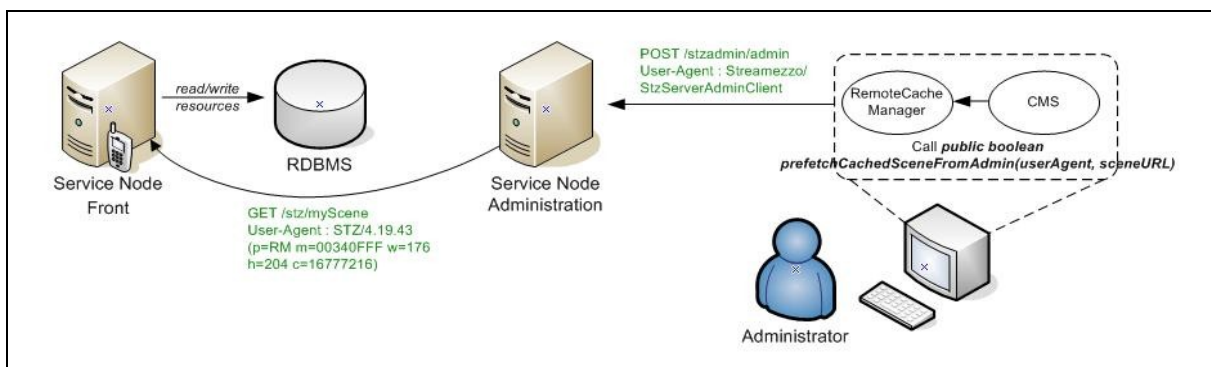
Note: In order to enable the pre-fetch for a service published onto the Rich Media Server, a URL mapping must be defined. For more information about URL mapping; please refer to the Rich Media Server Administration Guide.

There is two ways to simulate such a request. The first way consists in calling directly Rich Media Server Front with a request including a User-Agent header matching end user devices one.



This solution implies Rich Media Server Front is reachable from Content Management System.

The second way relies on Rich Media Server Administration which is in charge of requesting Rich Media Server Front. This way may be a workaround when Front is not reachable from CMS.



This solution implies Rich Media Server Front is reachable from Rich Media Server Administration.

Both ways allows forcing host and port to use as cached scene key. This is required when host and port used by devices cannot be used from backend. Such a case happens on most carrier-class network where mobile gateway is not reachable from backend LAN. This is the reason why many methods are provided for defining as many parameters as required.

At least, rich media scene URL mapping and simulated device User-Agent are supposed to be defined.

Please refer to StzManager API javadoc for more information about every method provided for prefetching a scene.