

Red5 Documentation

Daniel Rossi

Red5 Documentation

Daniel Rossi

Copyright © 2007 Daniel Rossi

Abstract

Table of Contents

.....	1
.....	2
Frequently Asked Questions	2
Project Management	6
Server Side Development	6
Codecs/Media integration	7
Client Side/API Testing	7
Branding/Logo/Website	7
Documentation	7
.....	9
How to build with eclipse	9
.....	10
.....	11
HOWTO create new applications in Red5	11
Preface	11
The application directory	11
Configuration	11
globalScope	11
contextConfigLocation	11
locatorFactorySelector	12
parentContextKey	12
log4jConfigLocation	12
webAppRootKey	12
Handler configuration	12
Context	12
Scopes	13
Handlers	13
Sample handler	14
.....	15
Preface	15
The application directory	15
Configuration	15
globalScope	15
contextConfigLocation	15
listener (start-up / shutdown)	16
parentContextKey	16
log4jConfigLocation	16
Handler configuration	16
Context	16
Scopes	16
Handlers	17
.....	18
HOWTO create new applications in Red5	18
Preface	18
The application directory	18
Configuration	18
globalScope	18
contextConfigLocation	18
locatorFactorySelector	19
parentContextKey	19
log4jConfigLocation	19
webAppRootKey	19
Handler configuration	19
Context	19
Scopes	20

Handlers	20
Sample handler	21
.....	22
HOWTO stream content to/from custom directories	22
Preface	22
Filename generator service	22
Custom generator	22
Activate custom generator	23
Change paths through configuration	23
.....	25
Security with Red5 0.6	25
Preface	25
Streams	25
Stream playback security	25
Stream publishing security	25
Shared objects	26
.....	28
JRuby	29
.....	29
application.rb - a translation into Ruby of the ofla demo application, a red5 example.....	29
.....	29
@author Paul Gregoire	29
.....	29
JRuby - style	31
.....	31
demoservice.rb - a translation into Ruby of the ofla demo application, a red5 example.....	31
.....	31
@author Paul Gregoire	31
.....	31
.....	35
.....	36
.....	37
Red5 release manual	37
.....	38
Specification of the RTMPT protocol	38
Overview	38
URLs	38
Request / Response	38
Polling interval	38
Initial connect (command “open”)	39
Client updates (command “send”)	39
Polling requests (command “idle”)	39
Disconnect of a session (command “close”)	39
.....	40
Red5 Changelog	40
Red5 0.6.4 (unreleased)	40
Red5 0.6.3 (2007-09-17)	40
Red5 0.6.2 (2007-06-17)	40
Red5 0.6.1 (2007-05-23)	41
Red5 0.6 (2007-04-23)	41
Red5 0.6rc3 (2007-04-11)	41
Red5 0.6rc2 (2007-02-12)	42
Red5 0.6rc1 (2006-10-30)	43
Red5 0.5 (2006-07-25)	43
Red5 0.5rc1 (2006-07-11)	43
Red5 0.4.1 (2006-05-01)	44
Red5 0.4 (2006-04-20)	44
Red5 0.3 (2006-02-21)	44

Red5 0.2 (2005–10–21)	44
-----------------------------	----

Welcome to the RED5 build, to build and run the RED5 server you will need to do the following :

1. Apache ANT version 1.6.2 download [here](#)_
2. Java JDK 1.5 download [here](#)_ You don't need netbeans unless you need an IDE for java
3. The RED5 distribution, download [here](#)_
4. Ant must be on your system path (test by typing `ant -version` at a system prompt)
5. You must have the environment variables for `JAVA_HOME` and `JAVA_VERSION` defined, typically

`JAVA_HOME=C:\sdk1.5.0_07` `JAVA_VERSION=1.5` You can check this on windows by typing `set JAVA_HOME` or on unix by typing `echo $JAVA_HOME`

FAILURE TO SETUP YOUR ENVIRONMENT VARIABLES WILL PREVENT YOUR FROM BEING ABLE TO BUILD PROPERLY

6. Now to just build the red5 server and start it running type `ant server` - This will compile the code and start the red5 server listening on port 1935.
7. To test the server use either the samples in the "swf" directory or write your own fla to call red5 using `rtmp://localhost:1935/oflaDemo` as the URI.
8. To change the default port Red5 listens on edit the `conf/red5.properties` and change the property `rtmp.host_port` value to your desired host and port i.e. : `rtmp.host_port = 0.0.0.0:2935` and then restart the server by typing

`ant server`

9. Please direct any questions to the red5 mailing list : red5@osflash.org

.._download here: <http://archive.apache.org/dist/ant/binaries/> .._download here: <http://java.sun.com/j2se/1.5.0/download.html> .._download here: <http://svn1.cvsdude.com/osflash/red5/>

.. image:: Frequently%20Asked%20Questions_html_5b618679.gif

.. image:: Frequently%20Asked%20Questions_html_11e056e5.png

=

Frequently Asked Questions

Table of Contents ~~~

- Downloading
- *Where can I download Red5?_*
- General
 - *What is Red5?_*
 - *Why does Red5 exist?_*
 - *What does Red5 mean?_*
 - *Why is it not called Red5 Media Server_*
 - *What other names does Red5 go by?_*
 - *What does the Red5 Logo look like_*
 - *RTMP (real time messaging protocol) has been introduced to the Flash Platform as a proprietary closed source protocol. Can we legally create source code that may unveil the true workings behind this protocol?_*
 - *What would be the best possible scenario that could come out of this project?_*
 - *What license does Red5 use?_*
 - *What is the estimated time of delivery for Red5 (eta)?_*
- Protocols
 - *RTMP_*
 - *AMF_*
- Server Development
 - *What implementations are currently being implemented?_*
 - *How far along is the Java implementation?_*
 - *Why did the Red5 team choose Mina?_*
 - *Why did the Red5 team choose Spring?_*
- Client Development
 - *Which component frameworks are we currently reviewing?_*
 - *What are the advantages of this framework?_*

-
- *What is the estimated time of delivery for these components (eta)?_*
 - *Do I have to use this component framework?_*
 - *Can I use Macromedia Flash Communication Server components?_*
 - Developer Resources
 - *What IDE (integrated development environment) are we using?_*
 - *Is there a place that we can meet and talk about the project status?_*
 - *Is there a mailing list ?_*
 - *Where's the best place to start?_*
 - *Who are the team members?_*
 - Accomplishments
 - *What are the milestones?_*
 - Monetary Support
 - *How can I donate to this project?_*
 - *How can my company help support this project?_*

Where can I download Red5? ~~~

Mirrors: http://osflash.org/red5#conference_links_

[Back to top_](#)

What is Red5? ~~~

An open source project dedicated towards the interaction between the Flash Player and a Free Connection Oriented Server using rtmp (real time messaging protocol).

[Back to top_](#)

Why does Red5 exist? ~~~

Like most open source projects, the project exists because there was interest in the topic. Even before any code was written people had been dissecting the bytes that come down the pipe from the flash comm server and flash player interaction.

[Back to top_](#)

What does Red5 mean? ~~~

Please read: *Why "RED5" ?_*

[Back to top_](#)

~

Why is it not called Red5 Media Server? ~~~

Red5 does much more than server up media. We feel that Red5 is a technology.

Back to top_

~

What other names does Red5 go by? ~~~

Red5 aka R5

Back to top_

What does the Red5 logo look like? ~~~

.. image:: Frequently%20Asked%20Questions_html_11e056e5.png_

Back to top_

RTMP (real time messaging protocol) has been introduced to the Flash Platform as a proprietary closed source protocol. Can we legally create source code that may unveil the true workings behind this protocol?

Yes, examining packets is not illegal. The Red5 team has stated its intent to remain completely legal in this situation, and thanks to some outstanding developers in the Flash world, we've never had to even consider any other alternative.

Back to top_

What would be the best possible scenario that could come out of this project?

Macromedia would endorse us and provide the open source community with the RTMP protocol specifications. This would greatly help out the few who are coding ?Free Servers? out there including the Red5 open source project.

Back to top_

What license does Red5 use? ~~~

Red5 uses the LGPL license.

LGPL license_

Back to top_

What is the estimated time of delivery for Red5 (eta) ~~~

Please Review: *Project Roadmap_*

Back to top_

RTMP ~~

Real Time Messaging Protocol (RTMP) Is a proprietary protocol developed by *Macromedia_* that is primarily used with *Macromedia_'s* Media Server product to stream audio and video over the web.

The default connection port is 1935.

wikipedia_

Red5 page dedicated to RTMP: http://osflash.org/rtmp_os_

Back to top_

AMF ~

Action Message Format (AMF) is a binary message format designed for the ActionScript object model.

Back to top_

What implementations are currently being implemented?

Java and Ruby. Currently all focus has been on the Java implementation.

Back to top_

How far along is the Java implementation?

There is a codebase checked into subversion. The code currently uses two frameworks, Mina and Spring discussed below. The protocol handler has been created and output of byte information is viewable. Team members are currently testing the code and sending different rtmp calls from the flash player to figure out how to deal with the rest of the rtmp protocol.

Back to top_

Why did the Red5 team choose Mina?

It allowed us to focus on the protocol, and not implementing low level nio code. We also plan to implement other protocols / transports in the future so having a standard framework is good.

I looked at a number of frameworks for this. Mina, EmberIO, Mule, etc. Mina seemed to be the most focused and developed (essentially being v2 of Netty). EmberIO is quite similar and something we should look into in the future, esp the threading strategies but not as mature or documented as a framework. Mule seems to be message exchange / network framework on speed. It does everything which I think is too much for our stage of development.

Back to top_

Why did the Red5 team choose Spring?

TODO

Back to top_

Which component frameworks are we currently reviewing?

Actionstep, ASwing

Back to top_

What are the advantages of this framework?

TODO

Back to top_

What is the estimated time of delivery for these components (eta)?

TODO

Back to top_

Do I have to use this component framework? ~~~~

TODO

Back to top_

Can I use Macromedia Flash Communication Server components? ~~~

TODO

Back to top_

What IDE (integrated development environment) are we using?

I'm using eclipse. I know a few others are too. As well, you will need to install the subversion plugin so that you can check out the code base. Additionally, you should install one of the actionsript plugins if you plan to do any of the front end coding.

Back to top_

Is there a place that we can meet and talk about the project status?

Yes, you can join a few of us developers on IRC (internet relay chat). Connect to the irc.freenet.org server and then join room #red5

Back to top_

Is there a mailing list ?

Yes, red5@osflash.org. You can find more information on the site <http://osflash.org/doku.php?id=red5>

Back to top_

Where's the best place to start?

Read about the protocol. <http://osflash.org/doku.php?id=red5#protocol>. Navigate the Red5 site, learn the basics behind the frameworks. Then check out the codebase and discuss through one of the many communication channels (irc, mailing list). If you are interested in becoming a Red5 team member please fill out the [Team Signup Form](#) and email it to the project managers.

Back to top_

What are the milestones?

TODO

Back to top_

Who are the team members?

Project Management

- *Chris Allen_*
- *John Grden_*

Server Side Development

- *Mick Merres_*
- *Luke Hubbard_*
- *Grant Davies_*
- *Fernando Augusto_*
- *Lucas Ferreira_*
- *Chris Allen_*

-
- *Ashvin Savani_*
 - *Yannick Connan_*
 - *Dominick Accattato_*

Release Manager ~

- *Grant Davies_*

Codecs/Media integration

- *Dominick Accattato_*

Client Side/API Testing

- *John Grden_*
- *Marcos Augusto_*
- *Gabriel Laet_*
- *Marlos Carmo_*
- *Tim Beynart_*
- *Lucas Ferreira_*
- *Chris Allen_*
- *Ashvin Savani_*

Branding/Logo/Website

- *Tim Beynart_*
- *Aldo Bucchi_*

Documentation

- *Patrick Mineault_* (AMF documentation)
- *John Grden_*
- *Lee McColl-Sylvester_*

Back to top_

Last revised: 10/24/05

Dominick Accattato (daccattato at gmail dot com)

.. image:: Frequently%20Asked%20Questions_html_m26c3cb3a.png

.. image:: Frequently%20Asked%20Questions_html_5b618679.gif

.. _Where can I download Red5?: #Where_can_I_download_Red5 .. _What is Red5?: #What_is_Red5 .. _Why does Red5 exist?: #Why_does_Red5_exist .. _What does Red5 mean?: #What_does_Red5_mean .. _Why is it not called Red5 Media Server: #Why_is_it_not_called_Red5_Media_Server .. _What

other names does Red5 go by?: #What_other_names_does_Red5_go_by ..
 _What does the Red5 Logo look like: #What_does_the_Red5_logo_look_like ..
 .. _RTMP (real time messaging protocol) has been introduced
 to the Flash Platform as a proprietary closed source protocol.
 Can we legally create source code that may unveil the
 true workings behind this protocol?: #RTMP_legal .. _What
 would be the best possible scenario that could come out
 of this project?: #Best_Scenario .. _What license does Red5
 use?: #What_license_does_Red5_use .. _What is the estimated time
 of delivery for Red5 (eta): #What_is_the_estimated_time_of_delivery ..
 _RTMP : #RTMP .. _AMF: #AMF .. _What implementations
 are currently being implemented?: #current_implementations .. _How far
 along is the Java implementation?: #current_java_status .. _Why
 did the Red5 team choose Mina?: #why_choose_mina .. _Why
 did the Red5 team choose Spring?: #Why_did_the_Red5_team_choose_Spring
 .. _Which component frameworks are we currently reviewing:
 #Which_component_frameworks_are_we .. _What are the advantages
 of this framework: #What_are_the_advantages_of_this .. _What is
 the estimated time of delivery for these components (eta):
 #clientside_component_eta .. _Do I have to use this component
 framework: #Do_I_have_to_use_this_component .. _Can I use Macromedia
 Flash Communication Server components: #Can_I_use_Macromedia_Flash_Communication
 .. _What IDE (integrated development environment) are we using?:
 #What_IDE .. _Is there a place that we can meet and
 talk about the project status?: #meeting_place .. _Is there
 a mailing list?: #is_there_a_mailing_list .. _Where's the best
 place to start?: #wheres_the_best_place_to_start .. _Who are the
 team members?: #team_members .. _What are the milestones:
 #What_are_the_milestones .. _How can I donate to this project:
 #How_can_I_donate_to_this_project .. _How can my company help
 support this project: #How_can_my_company_help_support .. http://osflash.org/red5#conference_links: http://osflash.org/red5#conference_links .. _Back to top: #top ..?:
http://osflash.org/red5#why_red5#whyred5 .. : <http://osflash.org/lib/exe/fetch.php?cache=cache&media=http%3A%2F%2Fwww.acmewebworks.com%2Ffred5%2Fimages%2FfinalLogo.png> .. _LGPL license: <http://www.opensource.org/licenses/lgpl-license.php> .. _Project Roadmap: http://review.codegent.net/opensource/RED5_draft_roadmap.pdf .. _Macromedia: <http://en.wikipedia.org/wiki/Macromedia>
 .. _wikipedia: http://en.wikipedia.org/wiki/Real_Time_Messaging_Protocol .. http://osflash.org/rtmp_os: http://osflash.org/rtmp_os .. red5@osflash.org: <mailto:red5@osflash.org> ..
<http://osflash.org/doku.php?id=red5>: <http://osflash.org/doku.php?id=red5> .. <http://osflash.org/doku.php?id=red5#protocol>: <http://osflash.org/doku.php?id=red5#protocol> .. _Team Signup Form:
http://osflash.org/red5_signupform .. _Chris Allen: http://osflash.org/chris_allen .. _John Grden: http://osflash.org/john_grden .. _Mick Merres: http://osflash.org/mick_merres .. _Luke Hubbard: http://osflash.org/luke_hubbard .. _Grant Davies: http://osflash.org/grant_davies .. _Fernando Augusto:
http://osflash.org/fernando_augusto .. _Lucas Ferreira: http://osflash.org/lucas_ferreira .. _Ashvin Savani:
http://osflash.org/ashvin_savani .. _Yannick Connan: http://osflash.org/yannick_connan ..
 _Dominick Accattato: http://osflash.org/dominick_accattato .. _Marcos Augusto: http://osflash.org/marcos_augusto .. _Gabriel Laet: http://osflash.org/gabriel_laet .. _Marlos Carmo: http://osflash.org/marlos_carmo .. _Tim Beynart: http://osflash.org/tim_beynart .. _Aldo Bucchi: http://osflash.org/aldo_bucchi .. _Patrick Mineault: http://osflash.org/patrick_mineault .. _Lee McColl-Sylvester: http://osflash.org/lee_mccoll-sylvester

.. image:: red5logo.png

How to build with eclipse

This guide assumes eclipse 3.1.0 and you have downloaded the entire red5 build from the subversion repository at http://svn1.cvsdude.com/osflash/red5_

1. Start Eclipse
2. From the File menu select “import”
3. In the Import dialog box select the item “Existing Projects into Workspace” and hit next
4. hit the “browse” button next to the “Select root directory” text box
5. select the root folder where you downloaded the red5 repository,(e.g. c:5) and hit ok
6. Make sure red5 is selected in the projects area and hit “Finish”
7. Eclipse should automatically build the project, you can force a build from the “project” menu and selecting “build project”
8. To run the server, in the package explorer panel, select the “Server” file under `src/org.red5.server/Standalone.java`, right click on the file and select “Run As” and select “Java Application”
9. Red 5 will start running

If you get an error in the console like :

```
java.net.BindException: Address already in use: bind at sun.nio.ch.Net.bind(Native
Method) at sun.nio.ch.ServerSocketChannelImpl.bind(Unknown Source) at
sun.nio.ch.ServerSocketAdaptor.bind(Unknown Source) at
org.apache.mina.io.socket.SocketAcceptor.registerNew(SocketAcceptor.java:362) at
org.apache.mina.io.socket.SocketAcceptor.access800(SocketAcceptor.java:46)atorg.apache.mina.io.socket.SocketAcce
Exception in thread “main”
```

Then the socket red5 wants to run is in use, you can change the socket and I will write this up later today once I speak with Luke.

.. _<http://svn1.cvsdude.com/osflash/red5>: <http://svn1.cvsdude.com/osflash/red5>

\$Date: \$ \$Revision: \$

Spring scripting support: <http://static.springframework.org/spring/docs/2.0.2/reference/dynamic-language.html#dynamic-language-resources> spring-support.jar

Groovy: <http://groovy.codehaus.org/> asm-2.2.2.jar antlr-2.7.6.jar groovy-1.0.jar

Beanshell: <http://www.beanshell.org/> bsh-2.0b5.jar cglib-nodep-2.1_3.jar

Ruby: <http://jruby.codehaus.org/> jruby.jar cglib-nodep-2.1_3.jar

Jython / Python: <http://www.jython.org/Project/index.html> jython.jar

The following are need for Java 5 only:

Script related JSR's: jsr-223-1.0-pr.jar jsr173_1.0_api.jar

Javascript / Rhino: <http://www.mozilla.org/rhino/> js.jar xbean.jar - needed for E4X

HOWTO create new applications in Red5

:Author: Joachim Bauch :Contact: jojo@struktur.de :Date: \$Date: 2006-04-27 07:45:42 +1000 (Thu, 27 Apr 2006) \$:Revision: \$Revision: 815 \$:Id: \$Id: HOWTO-NewApplications.txt 815 2006-04-26 21:45:42Z jbauch \$

.. contents::

Preface

This document describes how new applications can be created in Red5. It applies to the new API introduced by Red5 0.4.

The application directory

Red5 stores all application definitions as folders inside the “webapps” directory beneath the root of Red5. So the first thing you will have to do in order to create a new application, is to create a new subfolder in “webapps”. By convention this folder should get the same name the application will be reached later.

Inside your new application, you will need a folder “WEB-INF” containing configuration files about the classes to use. You can use the templates provided by Red5 in the folder “doc/templates/myapp”.

During the start of Red5, all folders inside “webapps” are searched for a directory “WEB-INF” containing the configuration files.

Configuration

The main configuration file that is loaded is “web.xml”. It contains the following parameters:

globalScope

The name of the global scope, this should be left at the default::

```
<context-param>
  <param-name>globalScope</param-name>
  <param-value>default</param-value>
</context-param>
```

contextConfigLocation

Specifies the name(s) of handler configuration files for this application. The handler configuration files reference the classes that are used to notify the application about joining / leaving clients and that provide the methods a client can call.

Additionally, the handler configuration files specify the scope hierarchy for these classes.

The path name given here can contain wildcards to load multiple files::

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/red5-*.xml</param-value>
</context-param>
```

locatorFactorySelector

References the configuration file of the root application context which usually is “red5.xml”::

```
<context-param>
  <param-name>locatorFactorySelector</param-name>
  <param-value>red5.xml</param-value>
</context-param>
```

parentContextKey

Name of the parent context, this usually is “default.context”::

```
<context-param>
  <param-name>parentContextKey</param-name>
  <param-value>default.context</param-value>
</context-param>
```

log4jConfigLocation

Path to the configuration file for the logging subsystem::

```
<context-param>
  <param-name>log4jConfigLocation</param-name>
  <param-value>/WEB-INF/log4j.properties</param-value>
</context-param>
```

webAppRootKey

Unique name for this application, should be the public name::

```
<context-param>
  <param-name>webAppRootKey</param-name>
  <param-value>/myapp</param-value>
</context-param>
```

Handler configuration

Every handler configuration file must contain at least three beans:

Context

The context bean has the reserved name `web.context` and is used to map paths to scopes, lookup services and handlers. The default class for this is `org.red5.server.Context`.

By default this bean is specified as::

```
<bean id="web.context" class="org.red5.server.Context"
      autowire="byType" />
```

Every application can only have one context. However this context can be shared across multiple scopes.

Scopes

Every application needs at least one scope that links the handler to the context and the server. The scopes can be used to build a tree where clients can connect to every node and share objects inside this scope (like shared objects or live streams). You can see the scopes as rooms or instances.

The default scope usually has the name `web.scope`, but the name can be chosen arbitrarily.

The bean has the following properties:

`server` This references the global server `red5.server`. `parent` References the parent for this scope and usually is `global.scope`. `context` The server context for this scope, use the `web.context` from above. `handler` The handler for this scope (see below). `contextPath` The path to use when connecting to this scope. `virtualHosts` A comma separated list of hostnames or ip addresses this scope runs at.

A sample definition looks like this::

```
<bean id="web.scope" class="org.red5.server.WebScope"
      init-method="register">
  <property name="server" ref="red5.server" />
  <property name="parent" ref="global.scope" />
  <property name="context" ref="web.context" />
  <property name="handler" ref="web.handler" />
  <property name="contextPath" value="/myapp" />
  <property name="virtualHosts" value="localhost, 127.0.0.1" />
</bean>
```

You can move the values for `contextPath` and `virtualHosts` to a separate properties file and use parameters. In that case you need another bean::

```
<bean id="placeholderConfig"
      class="org.springframework.beans.factory.config.PropertyPlaceholderConfig">
  <property name="location" value="/WEB-INF/red5-web.properties" />
</bean>
```

Assuming a `red5-web.properties` containing the following data::

```
webapp.contextPath=/myapp
webapp.virtualHosts=localhost, 127.0.0.1
```

the properties of the scope can now be changed to::

```
<property name="contextPath" value="${webapp.contextPath}" />
<property name="virtualHosts" value="${webapp.virtualHosts}" />
```

The `contextPath` specified in the configuration can be seen as “root” path of the scope. You can add additional elements after the configured path when connecting to dynamically create extra scopes.

These extra scopes all use the same handler but have their own properties, shared objects and live streams.

Handlers

Every context needs a handler that implements the methods called when a client connects to the scope, leaves it and that contains additional methods that can be called by the client. The interface

these handlers need to implement is specified by `org.red5.server.api.IScopeHandler`, however you can implement other interfaces if you want to control access to shared objects or streams.

A sample implementation that can be used as base class can be found at `org.red5.server.adapter.ApplicationAdapter`. Please refer to the javadoc documentation for further details.

The bean for a scope handler is configured by::

```
<bean id="web.handler"
      class="the.path.to.my.Application"
      singleton="true" />
```

The `id` attribute is referenced by the scope definition above.

If you don't need any special server-side logic, you can use the default application handler provided by Red5::

```
<bean id="web.handler"
      class="org.red5.server.adapter.ApplicationAdapter"
      singleton="true" />
```

Sample handler

A sample handler can be implemented in a few lines of code::

```
package the.path.to.my;

import org.red5.server.adapter.ApplicationAdapter;

public class Application extends ApplicationAdapter {

    public Double add(Double a, Double b){
        return a + b;
    }

}
```

Assuming the sample configuration above, you can call this method using the following ActionScript snippet::

```
nc = new NetConnection();
nc.connect("rtmp://localhost/myapp");
nc.onResult = function(obj) {
    trace("The result is " + obj);
}
nc.call("add", nc, 1, 2);
```

This should give you the output::

The result is 3

Preface

This document describes how applications can be configured in Red5 when using the WAR implementation. In this version of Red5 the J2EE container is not contained within Red5 and therefore is configured differently. This document assumes that the application WAR has already been expanded.

The application directory

An application war is normally expanded into a directory based upon the name of the war file, eg. red5.war expands into tomcat/webapps/red5 on a Tomcat server. In a standard Red5 installation, all the applications are stored within their own directory under the webapps directory; the difference here is that they are all located in the same directory.

Configuration

The WAR version stores all application definitions as Spring configuration files suffixed with the string “-context.xml”; If your application was called ofla then its configuration file would be named “ofla-context.xml”. The context files are loaded automatically upon server startup.

The main configuration file that is loaded is “web.xml”. It contains the following parameters:

globalScope

The name of the global scope, this should be left at the default::

```
<context-param>
  <param-name>globalScope</param-name>
  <param-value>default</param-value>
</context-param>
```

contextConfigLocation

Specifies the name(s) of handler configuration files for this application. The handler configuration files reference the classes that are used to notify the application about joining / leaving clients and that provide the methods a client can call.

Additionally, the handler configuration files specify the scope hierarchy for these classes.

The path name given here can contain wildcards to load multiple files::

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/applicationContext.xml, /WEB-INF/red5-common.xml, /WEB-INF/
</context-param>
```

listener (start-up / shutdown)

References the context listener servlet of the application, this technically takes the place of the Standalone.class in a standard Red5 server

```
<listener>
  <!-- Impersonates a org.springframework.web.context.ContextLoaderListener -
  <listener-class>org.red5.server.MainServlet</listener-class>
</listener>
```

parentContextKey

Name of the parent context, this usually is "default.context":

```
<context-param>
  <param-name>parentContextKey</param-name>
  <param-value>default.context</param-value>
</context-param>
```

log4jConfigLocation

Path to the configuration file for the logging subsystem:

```
<context-param>
  <param-name>log4jConfigLocation</param-name>
  <param-value>/WEB-INF/log4j.properties</param-value>
</context-param>
```

Handler configuration

Every handler configuration file must contain at least three beans:

Context

The default context bean has the reserved name "web.context" and is used to map paths to scopes, lookup services and handlers. The default class for this is "org.red5.server.Context".

By default this bean is specified as:

```
<bean id="web.context" class="org.red5.server.Context" autowire="byType" />
```

Every application can only have one context and they should follow this naming convention '`<application name>`'

.context' so that they will not conflict with one another. Application contexts can be shared across multiple scopes.

Scopes

Every application needs at least one scope that links the handler to the context and the server. The scopes can be used to build a tree where clients can connect to every node and share objects inside this

scope (like shared objects or live streams). You can see the scopes as rooms or instances. The default scope usually has the name “web.scope” and they should follow this naming convention '<application name>'

.scope' so that they will not conflict with one another.

The bean has the following properties:

“server” This references the global server `red5.server`. “parent” References the parent for this scope and usually is `global.scope`. “context” The server context for this scope, use the `web.context` from above. “handler” The handler for this scope (see below). “contextPath” The path to use when connecting to this scope. “virtualHosts” A comma separated list of hostnames or ip addresses this scope runs at. In this version we do not control the host names, this is accomplished by the server.

A sample definition looks like this::

```
<bean id="ofla.scope" class="org.red5.server.WebScope" init-method="register">
  <property name="server" ref="red5.server" />
  <property name="parent" ref="global.scope" />
  <property name="context" ref="ofla.context" />
  <property name="handler" ref="ofla.handler" />
  <property name="contextPath" value="/oflaDemo" />
  <property name="virtualHosts" value="localhost, 127.0.0.1" />
</bean>
```

The “contextPath” specified in the configuration can be seen as “root” path of the scope. You can add additional elements after the configured path when connecting to dynamically create extra scopes.

These extra scopes all use the same handler but have their own properties, shared objects and live streams.

Handlers

Every context needs a handler that implements the methods called when a client connects to the scope, leaves it and that contains additional methods that can be called by the client. The interface these handlers need to implement is specified by “org.red5.server.api.IScopeHandler”, however you can implement other interfaces if you want to control access to shared objects or streams.

A sample implementation that can be used as base class can be found at “org.red5.server.adapter.ApplicationAdapter”. Please refer to the javadoc documentation for further details.

The bean for a scope handler is configured by::

```
<bean id="ofla.handler" class="the.path.to.my.Application" singleton="true" />
```

The `id` attribute is referenced by the scope definition above.

If you don't need any special server-side logic, you can use the default application handler provided by Red5::

```
<bean id="web.handler" class="org.red5.server.adapter.ApplicationAdapter" single
```

HOWTO create new applications in Red5

:Author: Joachim Bauch :Contact: jojo@struktur.de :Date: \$Date: 2006-04-27 07:45:42 +1000 (Thu, 27 Apr 2006) \$:Revision: \$Revision: 815 \$:Id: \$Id: HOWTO-NewApplications.txt 815 2006-04-26 21:45:42Z jbauch \$

.. contents::

Preface

This document describes how new applications can be created in Red5. It applies to the new API introduced by Red5 0.4.

The application directory

Red5 stores all application definitions as folders inside the “webapps” directory beneath the root of Red5. So the first thing you will have to do in order to create a new application, is to create a new subfolder in “webapps”. By convention this folder should get the same name the application will be reached later.

Inside your new application, you will need a folder “WEB-INF” containing configuration files about the classes to use. You can use the templates provided by Red5 in the folder “doc/templates/myapp”.

During the start of Red5, all folders inside “webapps” are searched for a directory “WEB-INF” containing the configuration files.

Configuration

The main configuration file that is loaded is “web.xml”. It contains the following parameters:

globalScope

The name of the global scope, this should be left at the default::

```
<context-param>
  <param-name>globalScope</param-name>
  <param-value>default</param-value>
</context-param>
```

contextConfigLocation

Specifies the name(s) of handler configuration files for this application. The handler configuration files reference the classes that are used to notify the application about joining / leaving clients and that provide the methods a client can call.

Additionally, the handler configuration files specify the scope hierarchy for these classes.

The path name given here can contain wildcards to load multiple files::

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/red5-*.xml</param-value>
</context-param>
```

locatorFactorySelector

References the configuration file of the root application context which usually is “red5.xml”::

```
<context-param>
  <param-name>locatorFactorySelector</param-name>
  <param-value>red5.xml</param-value>
</context-param>
```

parentContextKey

Name of the parent context, this usually is “default.context”::

```
<context-param>
  <param-name>parentContextKey</param-name>
  <param-value>default.context</param-value>
</context-param>
```

log4jConfigLocation

Path to the configuration file for the logging subsystem::

```
<context-param>
  <param-name>log4jConfigLocation</param-name>
  <param-value>/WEB-INF/log4j.properties</param-value>
</context-param>
```

webAppRootKey

Unique name for this application, should be the public name::

```
<context-param>
  <param-name>webAppRootKey</param-name>
  <param-value>/myapp</param-value>
</context-param>
```

Handler configuration

Every handler configuration file must contain at least three beans:

Context

The context bean has the reserved name `web.context` and is used to map paths to scopes, lookup services and handlers. The default class for this is `org.red5.server.Context`.

By default this bean is specified as::

```
<bean id="web.context" class="org.red5.server.Context"
  autowire="byType" />
```

Every application can only have one context. However this context can be shared across multiple scopes.

Scopes

Every application needs at least one scope that links the handler to the context and the server. The scopes can be used to build a tree where clients can connect to every node and share objects inside this scope (like shared objects or live streams). You can see the scopes as rooms or instances.

The default scope usually has the name `web.scope`, but the name can be chosen arbitrarily.

The bean has the following properties:

`server` This references the global server `red5.server`. `parent` References the parent for this scope and usually is `global.scope`. `context` The server context for this scope, use the `web.context` from above. `handler` The handler for this scope (see below). `contextPath` The path to use when connecting to this scope. `virtualHosts` A comma separated list of hostnames or ip addresses this scope runs at.

A sample definition looks like this::

```
<bean id="web.scope" class="org.red5.server.WebScope"
      init-method="register">
  <property name="server" ref="red5.server" />
  <property name="parent" ref="global.scope" />
  <property name="context" ref="web.context" />
  <property name="handler" ref="web.handler" />
  <property name="contextPath" value="/myapp" />
  <property name="virtualHosts" value="localhost, 127.0.0.1" />
</bean>
```

You can move the values for `contextPath` and `virtualHosts` to a separate properties file and use parameters. In that case you need another bean::

```
<bean id="placeholderConfig"
      class="org.springframework.beans.factory.config.PropertyPlaceholderConfig">
  <property name="location" value="/WEB-INF/red5-web.properties" />
</bean>
```

Assuming a `red5-web.properties` containing the following data::

```
webapp.contextPath=/myapp
webapp.virtualHosts=localhost, 127.0.0.1
```

the properties of the scope can now be changed to::

```
<property name="contextPath" value="${webapp.contextPath}" />
<property name="virtualHosts" value="${webapp.virtualHosts}" />
```

The `contextPath` specified in the configuration can be seen as “root” path of the scope. You can add additional elements after the configured path when connecting to dynamically create extra scopes.

These extra scopes all use the same handler but have their own properties, shared objects and live streams.

Handlers

Every context needs a handler that implements the methods called when a client connects to the scope, leaves it and that contains additional methods that can be called by the client. The interface

these handlers need to implement is specified by `org.red5.server.api.IScopeHandler`, however you can implement other interfaces if you want to control access to shared objects or streams.

A sample implementation that can be used as base class can be found at `org.red5.server.adapter.ApplicationAdapter`. Please refer to the javadoc documentation for further details.

The bean for a scope handler is configured by::

```
<bean id="web.handler"
      class="the.path.to.my.Application"
      singleton="true" />
```

The `id` attribute is referenced by the scope definition above.

If you don't need any special server-side logic, you can use the default application handler provided by Red5::

```
<bean id="web.handler"
      class="org.red5.server.adapter.ApplicationAdapter"
      singleton="true" />
```

Sample handler

A sample handler can be implemented in a few lines of code::

```
package the.path.to.my;

import org.red5.server.adapter.ApplicationAdapter;

public class Application extends ApplicationAdapter {

    public Double add(Double a, Double b){
        return a + b;
    }

}
```

Assuming the sample configuration above, you can call this method using the following ActionScript snippet::

```
nc = new NetConnection();
nc.connect("rtmp://localhost/myapp");
nc.onResult = function(obj) {
    trace("The result is " + obj);
}
nc.call("add", nc, 1, 2);
```

This should give you the output::

The result is 3

HOWTO stream content to/from custom directories

:Author: Joachim Bauch :Contact: jojo@struktur.de :Date: \$Date: 2007-05-13 23:34:08 +1000 (Sun, 13 May 2007) \$:Revision: \$Revision: 2026 \$:Id: \$Id: HOWTO-StreamCustomDirectories.txt 2026 2007-05-13 13:34:08Z TTriemstra \$

.. contents::

Preface

This document describes how applications can stream ondemand videos (VOD) from or record to custom directories other than the default `streams` folder inside the webapp.

Filename generator service

Red5 uses a concept called `scope` services for functionality that is provided for a certain scope. One of these scope services is `IStreamFilenameGenerator_` that generates filenames for VOD streams that should be played or recorded.

Custom generator

To generate filename in different folders, a new filename generator must be implemented::

```
import org.red5.server.api.IScope;
import org.red5.server.api.stream.IStreamFilenameGenerator;

public class CustomFilenameGenerator implements IStreamFilenameGenerator {

    /** Path that will store recorded videos. */
    public String recordPath = "recordedStreams/";
    /** Path that contains VOD streams. */
    public String playbackPath = "videoStreams/";

    public String generateFilename(IScope scope, String name,
        GenerationType type) {
        // Generate filename without an extension.
        return generateFilename(scope, name, null, type);
    }

    public String generateFilename(IScope scope, String name,
        String extension, GenerationType type) {
        String filename;
        if (type == GenerationType.RECORD)
            filename = recordPath + name;
        else
            filename = playbackPath + name;

        if (extension != null)
            // Add extension
            filename += extension;
    }
}
```

```
        return filename;
    }
}
```

The above class will generate filenames for recorded streams like `recordedStreams/red5RecordDemo1234.flv` and use the directory `videoStreams` as source for all VOD streams.

Activate custom generator

In the next step, the custom generator must be activate in the configuration files for the desired application.

Add the following definition to `yourApp/WEB-INF/red5-web.xml`:

```
<bean id="streamFilenameGenerator"
      class="path.to.your.CustomFilenameGenerator" />
```

This will use the class defined above to generate stream filenames.

Change paths through configuration

While the class described here works as expected, it's a bit unhandy to change the paths inside the code as every change requires recompilation of the class.

Therefore you can pass parameters to the bean defined in the previous step to specify the paths to use inside the configuration file.

Add two methods to your class that will be executed while the configuration file is parsed::

```
public void setRecordPath(String path) {
    recordPath = path;
}

public void setPlaybackPath(String path) {
    playbackPath = path;
}
```

Now you can set the paths inside the bean definition::

```
<bean id="streamFilenameGenerator"
      class="path.to.your.CustomFilenameGenerator">
    <property name="recordPath" value="recordedStreams/" />
    <property name="playbackPath" value="videoStreams/" />
</bean>
```

You can also move the paths to the `yourApp/WEB-INF/red5-web.properties` file and use parameters to access them::

```
<bean id="streamFilenameGenerator"
      class="path.to.your.CustomFilenameGenerator">
    <property name="recordPath" value="${recordPath}" />
    <property name="playbackPath" value="${playbackPath}" />
</bean>
```

In that case you will have to add the following lines to your properties file::

```
recordPath=recordedStreams/  
playbackPath=videoStreams/
```

```
..    _IStreamFilenameGenerator:    http://dl.fancycode.com/red5/api/org/red5/server/api/stream/  
IStreamFilenameGenerator.html
```

Security with Red5 0.6

:author: Joachim Bauch :contact: jojo@struktur.de :Date: \$Date: 2007-03-30 08:33:46 +1000 (Fri, 30 Mar 2007) \$:Revision: \$Revision: 1798 \$:Id: HOWTO-Security.txt 1798 2007-03-29 22:33:46Z jbauch \$

.. contents::

Preface

This document describes the Red5 API that was introduced in version 0.6 to protect access to streams and/or shared objects similar to what the properties `Client.readAccess` and `Client.writeAccess` provide in the Macromedia Flash Communication Server / Flash Media Server 2.

Streams

Read (playback) and write (publishing/recording) access to streams is protected separately in Red5.

Stream playback security

For applications that want to limit the playback of streams per user or only want to provide access to streams with a given name, the interface `IStreamPlaybackSecurity_` is available in Red5.

It can be implemented by any object and registered in the `ApplicationAdapter_`. An arbitrary number of stream security handlers is supported per application. If at least one of the handlers denies access to the stream, the client receives an error `NetStream.Failed` with a `description` field giving a corresponding error message.

An example handler that only allows access to streams that have a name starting with `liveStream` is described below::

```
import org.red5.server.api.IScope; import org.red5.server.api.stream.IStreamPlaybackSecurity;

public class NamePlaybackSecurity implements IStreamPlaybackSecurity {

    public boolean isPlaybackAllowed(IScope scope, String name, int start,
        int length, boolean flushPlaylist) {
        if (!name.startsWith("liveStream")) {
            return false;
        } else {
            return true;
        }
    }
};
```

To register this handler in the application, add the following code in the `appStart` method::

```
registerStreamPlaybackSecurity(new NamePlaybackSecurity());
```

Red5 includes a sample security handler that denies all access to streams (`DenyAllStreamAccess_`).

Stream publishing security

In most applications that allow the user to publish and/or record streams, this access must be limited to prevent the server from being misused. Therefore, Red5 provides the interface `IStreamPublishSecurity_` to deny publishing of certain streams.

Similar to `IStreamPlaybackSecurity`, *it can be implemented by any object and registered in the `ApplicationAdapter`*. If one of the registered handlers denies access, the client receives an error `NetStream.Failed` with a `description` field giving a corresponding error message.

An example handler that only allows authenticated connections to publish a live stream starting with `liveStream` and deny all other access is described below::

```
import org.red5.server.api.IConnection;    import org.red5.server.api.IScope;    import
org.red5.server.api.Red5; import org.red5.server.api.stream.IStreamPublishSecurity;

public class AuthNamePublishSecurity implements IStreamPublishSecurity {

    public isPublishAllowed(IScope scope, String name, String mode) {
        if (!"live".equals(mode)) {
            // Not a live stream
            return false;
        }

        IConnection conn = Red5.getConnectionLocal();
        if (!"authenticated".equals(conn.getAttribute("UserType"))) {
            // User was not authenticated
            return false;
        }

        if (!name.startsWith("liveStream")) {
            return false;
        } else {
            return true;
        }
    }
};
```

To register this handler in the application, add the following code in the `appStart` method::

```
registerStreamPublishSecurity(new AuthNamePublishSecurity());
```

Of course, you will also have to add code in one of the `*Connect` or `*Join` methods that set the `UserType` attribute of a connection to `authenticated` for users that are allowed to publish streams.

Red5 includes a sample security handler that denies all access to streams (`DenyAllStreamAccess_`).

Shared objects

Once applications get complex, you might want to control the data that is stored in a shared object, thus not allowing the clients to modify SOs directly but only through methods exposed by the application.

The interface `ISharedObjectSecurity_` can be used to write handlers that deny certain actions on a given shared object or prevent the client from creating arbitrary shared objects.

Below is an example handler that only allows the creation of the persistent shared object `Chat`. Any client may connect to it and only sending messages `saySomething` through the SO is allowed. All write access to properties is denied. You could however change properties through serverside code as these changes are never protected by the security handlers.

::

```
import java.util.List; import org.red5.server.api.IScope; import org.red5.server.api.so.ISharedObject;
import org.red5.server.api.so.ISharedObjectSecurity;
```

```

public class SampleSOSecurityHandler implements ISharedObjectSecurity {

    public boolean isConnectionAllowed(ISharedObject so) {
        // Note: we don't check for the name here as only one SO can be
        //         created with this handler.
        return true;
    }

    public boolean isCreationAllowed(IScope scope, String name,
        boolean persistent) {
        if (!"Chat".equals(name) || !persistent) {
            return false;
        } else {
            return true;
        }
    }

    public boolean isDeleteAllowed(ISharedObject so, String key) {
        return false;
    }

    public boolean isSendAllowed(ISharedObject so, String message,
        List arguments) {
        if (!"saySomething".equals(message)) {
            return false;
        } else {
            return true;
        }
    }

    public boolean isWriteAllowed(ISharedObject so, String key,
        Object value) {
        return false;
    }
}

```

To register this handler in the application, add the following code in the `appStart` method::

```
registerSharedObjectSecurity(new SampleSOSecurityHandler());
```

If you want to register a security handler only for a given shared object, use code like this::

```

ISharedObject      so      =      getSharedObject(scope,      "MySharedObject");
so.registerSharedObjectSecurity(new MySOSecurityHandler());

..      _IStreamPlaybackSecurity:      http://dl.fancycode.com/red5/api/org/red5/server/api/stream/
IStreamPlaybackSecurity.html ..      _ApplicationAdapter:      http://dl.fancycode.com/red5/api/org/
red5/server/adapters/ApplicationAdapter.html ..      _DenyAllStreamAccess:      http://dl.fancycode.com/
red5/api/org/red5/server/api/stream/support/DenyAllStreamAccess.html ..      _IStreamPublishSecurity:
http://dl.fancycode.com/red5/api/org/red5/server/api/stream/IStreamPublishSecurity.html ..
_ISharedObjectSecurity:      http://dl.fancycode.com/red5/api/org/red5/server/api/so/
ISharedObjectSecurity.html

```

I. Select a scripting implementation Level: Beginner

Red5 includes interpreters for the following scripting languages: - Javascript - version 1.6 (Mozilla Rhino version 1.6 R7) - JRuby - version 1.0.1 (Ruby version 1.8.5) - Jython - version 2.2 (Python version 2.1) - Groovy - version 1.0 - Beanshell - version 2.0b4

Future versions may include: - JudoScript - Scala - PHP (This one is non-trivial, I may just provide a bridge) - Actionscript (Maybe SSAS)

The scripting implementation classes are pre-specified in the following locations depending upon your Java version:

Java5 - js-engine.jar, jython-engine.jar, groovy-engine.jar
Java6 - resources.jar

File location: /META-INF/services/javax.script.ScriptEngineFactory

It is most likely that the classes read from the jdk or jre will be preferred over any specified elsewhere.

II. Configuring Spring Level: Intermediate

Step one is to locate your web applications red5-web.xml file. Within the xml config file the web.scope bean definition must supply a web.handler, this handler is your Red5 application (An application must extend the org.red5.server.adapter.ApplicationAdapter class). The application provides access to the Red5 server and any service instances that are created. The service instances and the application itself may be scripted. Bean definitions in Spring config files may not have the same id, here are some web handler definition examples:

```
<bean></bean>
<bean> <constructor-arg index="0" value="classpath:applications/main.js"/> <constructor-arg
index="1">
<list>
<value> org.red5.server.api.IScopeHandler </value>
<value> org.red5.server.adapter.IApplication </value>
</list>
</constructor-arg> <constructor-arg index="2">
<value> org.red5.server.adapter.ApplicationAdapter </value>
</constructor-arg> </bean>
<bean> <constructor-arg index="0" value="classpath:applications/main.rb"/> <constructor-arg
index="1">
<list>
<value> org.red5.server.api.IScopeHandler </value>
<value> org.red5.server.adapter.IApplication </value>
</list>
</constructor-arg> </bean>
<bean> <constructor-arg index="0" value="classpath:applications/main.groovy"/> <constructor-arg
index="1">
<list>
<value> org.red5.server.api.IScopeHandler </value>
<value> org.red5.server.adapter.IApplication </value>
</list>
</constructor-arg> </bean>
<bean> <constructor-arg index="0" value="classpath:applications/main.py"/> <constructor-arg
index="1">
<list>
<value> org.red5.server.api.IScopeHandler </value>
<value> org.red5.server.adapter.IApplication </value>
</list>
```

```

</constructor-arg> <constructor-arg index="2">
<list>
<value> One </value>
<value> 2 </value>
<value> III </value>
</list>
</constructor-arg> </bean>

```

In general the configuration using scripted classes is defined using the constructor arguments (see interpreter section) in the following order: Argument 1 - Location of the script source file

Argument 2 - Java interfaces implemented by the script

The interfaces for the code which extends an Application are ba
have to use those interfaces in all your script definitions.

Argument 3 - Java classes extended by the script

The extended class is not always necessary, it depends upon the

The example location starts with classpath:applications which in physical disk terms for the "oflaDemo" application equates to webapps/oflaDemo/WEB-INF/applications

III.Creating an application script Level: Intermediate

1. Application adapter Scripting an application adapter is more difficult in some languages than it is in others, because of this I present the Ruby example which works really well and is easy to write and integrate. The application services are easily written in any of the supported languages, but they require a Java interface at a minimum.

- i. JRuby application adapter implementation -

JRuby

```

require "java" module RedFive include_package "org.red5.server.api" include_package
"org.red5.server.api.stream" include_package "org.red5.server.api.stream.support" include_package
"org.red5.server.adapter" include_package "org.red5.server.stream" end

```

application.rb - a translation into Ruby of the ofla demo application, a red5 example.

@author Paul Gregoire

```

class Application < RedFive::ApplicationAdapter

attr_reader :appScope, :serverStream
attr_writer :appScope, :serverStream

def initialize
  #call super to init the superclass, in this case a Java class
  super
  puts "Initializing ruby application"
end

```

```

def appStart(app)
  puts "Ruby appStart"
  @appScope = app
  return true
end

def appConnect(conn, params)
  puts "Ruby appConnect"
  measureBandwidth(conn)
  puts "Ruby appConnect 2"
  if conn.instance_of?(RedFive::IStreamCapableConnection)
    puts "Got stream capable connection"
    sbc = RedFive::SimpleBandwidthConfigure.new
    sbc.setMaxBurst(8388608)
    sbc.setBurst(8388608)
    sbc.setOverallBandwidth(8388608)
    conn.setBandwidthConfigure(sbc)
  end
  return super
end

def appDisconnect(conn)
  puts "Ruby appDisconnect"
  if appScope == conn.getScope && @serverStream != nil
    @serverStream.close
  end
  super
end

def toString
  return "Ruby toString"
end

def setScriptContext(scriptContext)
  puts "Ruby application setScriptContext"
end

def method_missing(m, *args)
  super unless @value.respond_to?(m)
  return @value.send(m, *args)
end
end

```

2. Application services Here is an example of a Java interface (Yes, the methods are supposed to be empty) which is used in the examples to provide a template for applications which will gather a list of files and return them as a “Map” (key-value pairs) to the caller.

i. Simple Java interface for implementation by scripts -

```

package org.red5.server.webapp.oflaDemo;

import java.util.Map;

public interface IDemoService {

  /**
   * Getter for property 'listOfAvailableFLVs'.

```

```

*
* @return Value for property 'listOfAvailableFLVs'.
*/
public Map getListOfAvailableFLVs();

public Map getListOfAvailableFLVs(String string);
}

```

ii. Spring bean definition for a script implementation of the interface -

<bean>

<constructor-arg index="0" value="classpath:applications/demoservice.rb"/> <constructor-arg index="1">

<list>

<value>

org.red5.server.webapp.oflaDemo.IDemoService

</value>

</list>

</constructor-arg>

</bean>

iii. JRuby script implementing the interface -

JRuby - style

```

require "java" module RedFive include_package "org.springframework.core.io" include_package
"org.red5.server.webapp.oflaDemo" end include_class "org.red5.server.api.Red5" include_class
"java.util.HashMap"

```

demoservice.rb - a translation into Ruby of the ofla demo application, a red5 example.

@author Paul Gregoire

```

class DemoService < RedFive::DemoServiceImpl

  attr_reader :filesMap
  attr_writer :filesMap

  def initialize
    puts "Initializing ruby demoservice"
    super
    @filesMap = HashMap.new
  end

  def getListOfAvailableFLVs
    puts "Getting the FLV files"
    begin

```

```

    dirname = File.expand_path('webapps/oflaDemo/streams').to_s
    Dir.open(dirname).entries.grep(/\.flv$/) do |dir|
      dir.each do |flvName|
        fileInfo = Hash.new
        stats = File.stat(dirname+'/'+flvName)
        fileInfo["name"] = flvName
        fileInfo["lastModified"] = stats.mtime
        fileInfo["size"] = stats.size || 0
        @filesMap[flvName] = fileInfo
        print 'FLV Name:', flvName
        print 'Last modified date:', stats.mtime
        print 'Size:', stats.size || 0
        print '-----'
      end
    end
  rescue Exception => ex
    puts "Error in getListOfAvailableFLVs #{errorType} \n"
    puts "Exception: #{ex} \n"
    puts caller.join("\n");
  end
  return filesMap
end

def formatDate(date)
  return date.strftime("%d/%m/%Y %I:%M:%S")
end

def method_missing(m, *args)
  super unless @value.respond_to?(m)
  return @value.send(m, *args)
end

end

```

iv. Java application implementing the interface, upon which the Ruby code was based (This code is NOT needed when using the script) -

```

package org.red5.server.webapp.oflaDemo;

import java.io.File; import java.io.IOException; import java.text.SimpleDateFormat; import
java.util.Date; import java.util.HashMap; import java.util.Locale; import java.util.Map;

import org.apache.commons.logging.Log; import org.apache.commons.logging.LogFactory;
import org.red5.server.api.IScope; import org.red5.server.api.Red5; import
org.springframework.core.io.Resource;

public class DemoService {

protected static Log log = LogFactory.getLog(DemoService.class.getName());

/**
 * Getter for property 'listOfAvailableFLVs'.
 *
 * @return Value for property 'listOfAvailableFLVs'.
 */
public Map getListOfAvailableFLVs() {
  IScope scope = Red5.getConnectionLocal().getScope();
  Map<String, Map> filesMap = new HashMap<String, Map>();
  Map<String, Object> fileInfo;

```

```

try {
    log.debug("getting the FLV files");
    Resource[] flvs = scope.getResources("streams/*.flv");
    if (flvs != null) {
        for (Resource flv : flvs) {
            File file = flv.getFile();
            Date lastModifiedDate = new Date(file.lastModified());
            String lastModified = formatDate(lastModifiedDate);
            String flvName = flv.getFile().getName();
            String flvBytes = Long.toString(file.length());
            if (log.isDebugEnabled()) {
                log.debug("flvName: " + flvName);
                log.debug("lastModified date: " + lastModified);
                log.debug("flvBytes: " + flvBytes);
                log.debug("-----");
            }
            fileInfo = new HashMap<String, Object>();
            fileInfo.put("name", flvName);
            fileInfo.put("lastModified", lastModified);
            fileInfo.put("size", flvBytes);
            filesMap.put(flvName, fileInfo);
        }
    }

    Resource[] mp3s = scope.getResources("streams/*.mp3");
    if (mp3s != null) {
        for (Resource mp3 : mp3s) {
            File file = mp3.getFile();
            Date lastModifiedDate = new Date(file.lastModified());
            String lastModified = formatDate(lastModifiedDate);
            String flvName = mp3.getFile().getName();
            String flvBytes = Long.toString(file.length());
            if (log.isDebugEnabled()) {
                log.debug("flvName: " + flvName);
                log.debug("lastModified date: " + lastModified);
                log.debug("flvBytes: " + flvBytes);
                log.debug("-----");
            }
            fileInfo = new HashMap<String, Object>();
            fileInfo.put("name", flvName);
            fileInfo.put("lastModified", lastModified);
            fileInfo.put("size", flvBytes);
            filesMap.put(flvName, fileInfo);
        }
    }
} catch (IOException e) {
    log.error(e);
}
return filesMap;
}

private String formatDate(Date date) {
    SimpleDateFormat formatter;
    String pattern = "dd/MM/yy H:mm:ss";
    Locale locale = new Locale("en", "US");
    formatter = new SimpleDateFormat(pattern, locale);
    return formatter.format(date);
}

```

```
}
```

v. Flex AS3 method calling the service -

```
[Bindable] public var videoList:ArrayCollection;
```

```
public function catchVideos():void{ // call server-side method // create a responder and set it to  
getMediaList var nc_responder:Responder = new Responder(getMediaList, null); // call the server  
side method to get list of FLV's nc.call("demoService.getListOfAvailableFLVs", nc_responder); }
```

```
public function getMediaList(list:Object):void{ // this is the result of the server side  
getListOfAvailableFLVs var mediaList:Array = new Array(); for(var items:String in list){  
mediaList.push({label:items, size:list[items].size, dateModified:list[items].lastModified}); } //  
videoList is bindable and the datagrid is set to use this for it's dataprovider // wrap it in an  
ArrayCollection first videoList = new ArrayCollection(mediaList); }
```

IV.Creating your own interpreter Level: Advanced

Lets just open this up by saying that I attempted to build an interpreter for PHP this last weekend 02/2007 and it was a real pain; after four hours I had to give up. So what I learned from this is that you must first identify scripting languages which operate as applications, not as http request processors. Heres a test: Can X language be compiled into an executable or be run on the command-line? If yes then it should be trivial to integrate.

V. Links with scripting information Spring scripting - <http://static.springframework.org/spring/docs/2.0.x/reference/dynamic-language.html> <http://rhinoinspring.sourceforge.net/>

Java scripting - <http://java.sun.com/developer/technicalArticles/J2SE/Desktop/scripting/> <http://blogs.sun.com/sundararajan/> <https://scripting.dev.java.net/> <http://today.java.net/pub/a/today/2006/04/11/scripting-for-java-platform.html> http://www.javaworld.com/javaworld/jw-03-2005/jw-0314-scripting_p.html http://www.oreillyn.com/onjava/blog/2004/01/java_scripting_half_the_size_h.html <http://www.robert-tolksdorf.de/vmlanguages.html>

Javascript - <http://www.mozilla.org/rhino/> <http://www.mozilla.org/rhino/ScriptingJava.html>

Ruby - <http://jruby.codehaus.org/>

BeanShell - <http://www.beanshell.org/>

Python - <http://www.jython.org/Project/> <http://www.onjava.com/pub/a/onjava/2002/03/27/jython.html> <http://jepp.sourceforge.net/> <http://jpe.sourceforge.net/> <http://jpye.sourceforge.net/>

Groovy - <http://groovy.codehaus.org/>

I. JMX classes Red5's implementation consists of the following classes and various other MBeans:

org.red5.server.jmx.JMXFactory - Provides access to the platform MBeanServer as well as registration, unregistration, and creation of new MBean instances. Creation and registration is performed using StandardMBean wrappers.

org.red5.server.jmx.JMXAgent - Provides the HTML adapter and registration of MBeans.

org.red5.server.jmx.JMXUtil - Helper methods for working with ObjectName or MBean instances.

II. Spring configuration The Spring configuration for the JMX implementation allows you to configure the “domain” for MBean registration and listener port for the HTML adaptor. The default entries are shown below.

```
<bean> </bean>
<bean> </bean>
```

The HTML adapter is disabled by default, but it allows easy management of MBeans from a web browser. The RMI adapter may only be used if an RMI registry is running. To start a registry simply execute this at the command prompt:

- windows rmiregistry 9999
- unix rmiregistry 9999 &

The “9999” is the port and should match your RMI configuration.

III.jConsole

JConsole is a utility that ships with the JRE (since 1.5), it allows you to manage local and remote JMX implementations. To enable introspection you must add the following VM parameter to your startup:

```
-Dcom.sun.management.jmxremote
```

After the parameter is set and the application initialized you can start jConsole at the command line by typing:

```
jconsole
```

A Swing application will appear and you must select the implementation (agent) you wish to manage, for local simply select “org.red5.server.Standalone”.

IV.Links <http://www.onjava.com/pub/a/onjava/2004/09/29/tigerjmx.html?page=1> <http://java.sun.com/developer/WDCTechTips/2005/tt0315.html#2>

===== HOWTO build a Red5 debian package
=====

1. Install the debian packages “dpkg-dev”, “debhelper”, “dh-make”, “devscripts” and “fakeroot”.
2. Checkout the debian build scripts to a folder “debian” inside the Red5 root from <http://svn1.cvsdude.com/osflash/red5/debian/trunk>
3. Update “debian/changelog” and add an entry for the new version you are building. Note that the syntax must match the previous entries!
4. Update the filename in “debian/files” to match the version you are building.
5. Make sure you run from a clean Red5 checkout of the tag to build!!!
6. From the red5 root run “dpkg-buildpackage -uc -b -rfakeroot”
7. If all goes well, you should have a debian package one folder below the Red5 root.

Red5 release manual

This document describes the steps necessary to create a new release of Red5:

1. Make sure everything has been committed to the trunk or correct branch.
2. Update the file doc/changelog.txt with informations about the new release.
3. Create tags of the modules that are linked into the main code tree:
 - documentation at <http://svn1.cvsdude.com/osflash/red5/doc/tags>

Tags for versions should always be the version string with dots replaced by underscores, e.g. version “1.2.3” becomes tag “1_2_3”.

If you would tag the documentation folder for version “1.2.3”, you would use the url http://svn1.cvsdude.com/osflash/red5/doc/tags/1_2_3

4. Tag the server code according to the naming scheme from above at <http://svn1.cvsdude.com/osflash/red5/java/server/tags>
5. Update the svn:externals in the newly created server code tag to point to the tagged modules from step 3.
6. You're done.

Specification of the RTMPT protocol

:author: Joachim Bauch :contact: mail@joachim-bauch.de :date: 2006-03-23 :copyright: Creative Commons License (by-sa)___

___ <http://creativecommons.org/licenses/by-sa/2.5/>

Overview

This document describes the RTMPT tunneling protocol as implemented by the Red5 Open Source Flash Server. Please note that this document is *not* an official specification by Macromedia but hopefully helps other people to write software that makes use of RTMPT.

RTMPT basically is a HTTP wrapper around the RTMP protocol that is sent using POST requests from the client to the server. Because of the non-persistent nature of HTTP connections, RTMPT requires the clients to poll for updates periodically in order to get notified about events that are generated by the server or other clients.

During the lifetime of a RTMPT session, four possible request types can be sent to the server which will be described below.

URLs

The URL to be opened has the following form::

```
http://server/<comand>/[<client>/]<index>
```

<command> denotes the RTMPT request type (see below) <client> specifies the id of the client that performs the requests (only sent for established sessions) <index> is a consecutive number that seems to be used to detect missing packages

Request / Response

All HTTP requests share some common properties:

- They use HTTP 1.1 POST.
- The content type is `application/x-fcs`.
- The connection should be kept alive by the client and server to reduce network overhead.

The HTTP responses also share some properties:

- The content type is `application/x-fcs`.
- For all established sessions the first byte of the response data controls the polling interval of the client where higher values mean less polling requests.

Polling interval

The server always starts with a value of 0x01 after data was returned and increases it after 10 empty replies. The maximum delay is 0x21 which causes a delay of approximately 0.5 seconds between two requests.

Red5 currently increases the delay in the following steps: 0x01, 0x03, 0x05, 0x09, 0x11, 0x21.

Initial connect (command “open”)

This is the first request that is sent to the server in order to register a client on the server and start a new session. The server replies with a unique id (usually a number) that is used by the client for all future requests.

Note: the reply doesn't contain a value for the polling interval! A successful connect resets the consecutive index that is used in the URLs.

Client updates (command “send”)

The data a client would send to the server using RTMP is simply prefixed with a HTTP header and otherwise sent unmodified.

The server responds with a HTTP response containing one byte controlling the polling interval and the RTMP data if available.

Polling requests (command “idle”)

If the client doesn't have more data to send to the server, he has to poll for updates to receive streaming data or events like shared objects.

Disconnect of a session (command “close”)

If a client wants to terminate his connection, he sends the “close” command which is replied with a 0x00 by the server.

Red5 Changelog

This file contains informations about the changes between the different versions of Red5.

Red5 0.6.4 (unreleased)

New Features: - Added stream listeners that can get notified about received packets
- Support for server-side Javascript (Jira APPSERVER-169) - Added new base class `org.red5.server.adapter.MultiThreadedApplicationAdapter` that allows multiple clients to connect simultaneously to the same application

Bugfixes: - Pause near end of buffered streams works as expected (Jira APPSERVER-199) - Fixed potential memory leak with RTMPT connections that are not properly closed (Jira APPSERVER-193)
- “onMetaData” is only written to newly recorded FLV files and contains valid properties now - Don't try to decode objects for closed RTMPT connections (Jira APPSERVER-208) - New multi-threaded connection code fixes various timeout issues (Jira APPSERVER-122, Jira APPSERVER-166 and Jira APPSERVER-167) - Always use correct classloader inside applications (Jira APPSERVER-200)
- Tomcat cannot undeploy red5 application (Jira APPSERVER-204) - “ByteArray” objects used old data after calling “compress” or “uncompress” (Jira APPSERVER-211) - “@DontSerialize” checks for properties also in inherited classes (Jira APPSERVER-225)

Red5 0.6.3 (2007-09-17)

New Features: - Remoting requests from “mx:RemoteObject” supported (Jira APPSERVER-144) - RTMPT working with Tomcat - Added thread that writes modified persistent objects periodically. This reduces server load if multiple attributes of one object, or the same object is modified frequently.
- Location of “webapps” folder can be configured in bean “jetty6.server” inside “conf/red5.xml” (Jira APPSERVER-152) - “IStreamFilenameGenerator” can specify if it returns absolute or relative paths
- Applications can be unloaded and loaded without restarting Red5 - “mx.collections.ArrayCollection” objects supported by AMF3 codec - Object attributes are converted if necessary in AMF0/AMF3 codecs - “mx.utils.ObjectProxy” objects supported by AMF3 codec (Jira APPSERVER-173) - “IConnection” objects for Remoting properly store attributes accross multiple requests by using sessions - Remoting headers are accessible through “IConnection.getConnectionParams” - “ByteArray” objects supported (Jira APPSERVER-189) - “NetStream.send” messages are properly passed through from Flex clients (Jira APPSERVER-185) - Class fields that should not be serialized when sending objects to clients can be annotated with “@DontSerialize” (in “org.red5.annotations”) - Public methods can be protected from being called through RTMP, RTMPT or Remoting by using “@DeclarePrivate” and “@DeclareProtected”. - Support for XML objects added to AMF3 codec (Jira APPSERVER-196)

Bugfixes: - Validate RTMP handshake received from client (Jira APPSERVER-159) - Array typed parameters are converted correctly (Jira APPSERVER-161) - RTMPHandler is wired through Spring (Jira APPSERVER-150) - fixed concurrency issue in RTMP encoder that could result in wrong packet header types (Jira APPSERVER-177) - IStreamAwareScopeHandler methods are also called for server side streams - “NetConnection.Connect.AppShutdown” is returned when trying to connect to application that currently is unloaded (Jira APPSERVER-13) - State is properly reset if exceptions occur in package decoding (Jira APPSERVER-137) - Numbers outside integer range are correctly serialized in AMF3 codec - return proper error object that triggers “onStatus” for “NetConnection.call” in case of errors (Jira APPSERVER-192) - Fixed endless loop in playlist controller with only one item in it (Jira APPSERVER-191) - Fixed renaming across filesystems (Jira SN-59) - Updated Jetty to 6.1.5 (Jira APPSERVER-123) - Fixed deserialization of AMF3 encoded SO events (Jira APPSERVER-188)

Red5 0.6.2 (2007-06-17)

Bugfixes: - “pause” no longer breaks live streams (Jira APPSERVER-136) - Configured subscopes don't get released when a client disconnects - AMF requests could not be decoded when run in

the context root (Jira APPSERVER-146) - Fixed bug for Remoting requests without parameters (Jira APPSERVER-147) - Fixed issue with stop/start of war in Tomcat (Jira APPSERVER-155) - Fixed handshake reply for Flash Player 9 Update 3 - IMetaData supports fractional framerates (Jira APPSERVER-157) - Correctly reject empty stream names (Jira APPSERVER-156) - Fixed problem with loading some JAR files from the applications classpath (Jira APPSERVER-141) - Fixed decoding of Remoting requests with multiple parameters (Jira APPSERVER-151)

Red5 0.6.1 (2007-05-23)

New Features: - Switched to use mina 1.1, more config options in red5.properties - Newly recorded files start with an "onMetaData" tag containing the duration and the codecs used - Added a JMX subsystem with RMI and HTTP connectors - Simplified MBean unregistration and added a registration check prior to the unregister attempt (Jira APPSERVER-118) - "IServerStream" now also supports "pause" and "seek" - Enabled RMI + SSL for JMX - Added JMX authentication - Added Shutdown class for cleanly shutting down a Red5 instance - Added support for AMF3 in remoting server - "receiveAudio" and "receiveVideo" work for VOD streams (Jira SN-22)

Bugfixes: - "NetStream.Record.Failed" is sent for IO errors that occurred during recording (Jira APPSERVER-64) - Fixed possible deadlock if methods are invoked by a connecting client on a client that is currently disconnecting (Jira APPSERVER-108) - Fixed NPE when connecting without application given (Jira APPSERVER-116) - Fixed various problems with deserialization of AMF3 objects that implement IExternalizable (Jira CODECS-2) - Fixed warning about deprecated Jetty configuration (Jira APPSERVER-115) - Fixed possible deadlock involving PersistableAttributeStore and Scope (Jira APPSERVER-122) - Display better message if RMI connection to "rmiregistry" could not be established (Jira APPSERVER-125) - Python scripts can import classes available only in the classpath of a webapp (Jira APPSERVER-92) - Fixed Ruby application issue by updating to Spring 2.0.5 and JRuby 0.9.8 (Jira APPSERVER-93) - Fixed async calling of remoting methods (Jira APPSERVER-131) - Accessing root of RTMPT server no longer results in 404 but redirects to HTTP port (Jira APPSERVER-130) - Disconnect clients that don't send a valid handshake (Jira APPSERVER-128) - Reduced max. idle time to prevent too many open sockets when using RTMPT with HTTP/1.0 (Jira APPSERVER-87) - Fixed potential NPEs in PlaylistSubscriberStream (Jira SN-40) - Fixed various problems with deserializing AMF0 references in remoting - Fixed frozen video if audio is disabled in live streams (Jira SN-22)

Red5 0.6 (2007-04-23)

New features: - Recording/playback of files to/from subscores implemented (Jira APPSERVER-103)

Bugfixes: - Ghost connection detection code rewritten to better detect dead clients (Jira APPSERVER-38, SN-37) - Deserialization of objects defined in webapp classpath fixed (Jira APPSERVER-80, APPSERVER-100) - Fixed AMF3 deserializer for references from attributes to parent classes (Jira APPSERVER-101) - Jython example adjusted for new bandwidth API (Jira APPSERVER-92) - Workaround added to deal with broken MP3 files (Jira APPSERVER-62) - "start" and "length" are properly evaluated when playing back VOD streams - Fixed seeking not working for MP3 or audio-only FLV files - Don't log contents of wrong objects (Jira APPSERVER-109) - Fixed potential NPEs in PlaylistSubscriberStream - A client buffer of 0 on live streams no longer breaks playback (Jira CS-3) - Fixed shutdown error in Tomcat with WAR version by updating to SLF4J 1.3.1 (Jira APPSERVER-107) - "NetStream.Play.InsufficientBW" is sent if client is too slow receiving video streams (Jira APPSERVER-51) - Improved frame dropping code for slow connections

Red5 0.6rc3 (2007-04-11)

New features: - Keyframe informations are cached so files don't need to be reparsed before playback - Connections from Flash Media Encoder and On2 Flix Live supported - Access to shared objects can be limited (Jira APPSERVER-25) - Connections can provide a list of remote addresses. This is usefull for proxied RTMPT connections.

Bugfixes: - Bandwidth control code has been rewritten to fix stability issues and memory leaking in high concurrency connection count situations - Serialization of Maps with non-number keys fixed (Jira APPSERVER-60) - Multiple IO processor threads are used by default - Memory leak when closing RTMPT connections fixed (Jira APPSERVER-61) - Merged WAR build script with primary script, also moved WAR specific startup servlet into trunk - Deserializing of remoting results fixed (Jira APPSERVER-63) - Fixed “error in object encoding” when rejecting AMF3 clients (Jira APPSERVER-73) - Concurrency problems when closing a connection fixed (Jira APPSERVER-59) - Unnecessary NetStream.Play.* events are no longer sent when playback stopped (Jira APPSERVER-70) - SimplePlaylistController setRepeat and setRandom fixed (Jira SN-27) - NPE in SimpleBWControlService fixed (Jira APPSERVER-75) - Reference bugs in AMF3 encoder fixed (Jira APPSERVER-81) - “NetStream.Play.Failed” is sent correctly now (Jira APPSERVER-52) - Concurrency issue fixed in SimpleBWControlService (Jira SN-32) - Fixed problem when decoding MP3 files with signed values in the ID3v2 tag size (Jira APPSERVER-86) - “NetStream.Seek.Failed” is sent when trying to seek in live streams (Jira APPSERVER-84) - “NetStream.Failed” is sent for exceptions during streaming methods (Jira APPSERVER-85) - Random server freezing resolved (Jira APPSERVER-41) - Send correct timestamps if seeking beyond end of file (Jira APPSERVER-54) - Fixed NoSuchElementException when iterating connections during disconnect (Jira APPSERVER-94) - Reference bugs in AMF3 decoder fixed (Jira APPSERVER-95) - “NetStream.Play.Complete” is sent (APPSERVER-50) - “NetStream.Play.Switch” is sent (APPSERVER-82) - Streams are always played to the end (SN-8) - Seeking in stopped streams fixed (APPSERVER-89) - Fixed deadlock in shared objects under high load (APPSERVER-98)

Red5 0.6rc2 (2007-02-12)

New features: - Stream classes can be configured through red5-common.xml (Trac #223) - RTMP network library supports client mode (Trac #94) - Source of VOD streams can be customized through IStreamFilenameGenerator (Trac #120) - API: IStreamFilenameGenerator differs between playback and recording - Results of method calls can be deferred until they are available to free io threads - Transient fields will not be serialized any longer (Jira APPSERVER-27) - Red5 compiles with Java6 now - Support for AMF3 incl. IExternalizable objects added (Jira APPSERVER-31) - Access to streams can be limited (Jira APPSERVER-25) - (non-persistent) shared objects can be acquired by serverside code to prevent them from being released when the last client disconnects (Jira APPSERVER-48)

Bugfixes: - Serialize RecordSet objects (Trac #201) - “NetConnection.Connect.Rejected” is sent for non-existing scopes to match result code of FCS/FMS - RTMPT through Jetty working again (Trac #213) - Size of last frame is correctly written to .flv files - Errors during “connect” are reported back to client through RTMPT - Fixed NPE in FlowControlService thread (Trac #175) - Deserializing of mixed arrays now works in all cases (Trac #109, #195) - “NetStream.Record.Start” and “NetStream.Record.Stop” are sent (Trac #127) - “NetStream.Publish.BadName” is sent if two clients try to publish/record a stream with the same name - Streams stopped if bandwidth limit was set too high (Trac #165) - Fixed potential concurrency issue in FlowControlService (Trac #224) - Stream notification callbacks are invoked on reused connections (Trac #133) - The playlist is flushed by default (Jira APPSERVER-6) - Fixed ClassCastException in “pendingVideoMessages” (Jira APPSERVER-14) - calling “pause” with null argument works again (Jira APPSERVER-12) - “NetStream.Publish.BadName” is only sent if another client is already publishing a stream - Playing a stream while being recorded now works (Jira SN-4, SN-13) - “IPendingServiceCall.isSuccess()” returns true when a result has been received (Jira APPSERVER-35) - The “http.host” setting from “red5.properties” is evaluated (Jira APPSERVER-36) - “IBroadcastStream” knows about the filename it is being recorded to (Jira APPSERVER-30) - BufferOverflowException for empty RTMP packets fixed (Jira APPSERVER-37) - FLV files are no longer locked after playback (Jira APPSERVER-17) - SharedObjects support “getAttributes” (Jira APPSERVER-45) - MP3 files containing images can be played back (Jira APPSERVER-47) - Fixed parsing of long strings (Jira APPSERVER-44) - Fixed pausing and seeking audio-only flv files (Jira SN-17) - Number of streams is no longer limited (Jira SN-14) - “NetStream.Play.Failed” is returned if a VOD stream can not be played due to IO errors (Jira APPSERVER-52) - “NetStream.InvalidArg” is returned for invalid arguments (Jira APPSERVER-55) - “NetConnection.Connect.InvalidApp” is returned for non-existing application

scopes on the server - “NetStream.Record.NoAccess” is returned if file could not be created or written to (Jira APPSERVER-53) - Error when setting SO attributes fixed (Jira APPSERVER-57)

Red5 0.6rc1 (2006-10-30)

New features: - Created WAR (Web Application Archive) version of Red5 (Separate repository java/war) - Enabled Tomcat or Jetty as J2EE container implementations - FLV cache implementations (2 are included) (Trac #99) - Scripting support (javascript, ruby, python, groovy, and bsh) based on Spring 2 and JSR223

Bugfixes: - Last frames aren't lost when reading .flv files (Trac #90) - FileConsumer acted on all consumer pipe events (Trac #92) - Improved timestamps of live streams to be more in sync with FMS (Trac #93) - FileConsumer modified position of incoming messages (Trac #91) - Events should support reference counting (Trac #103) - ServerStream playback jerky (Trac #77) - “NetStream.send” events are properly recorded - Reusing streams works (Trac #123) - Fixed NPE if no bandwidth settings are available (Trac #129) - “close” can be called on RTMPT connections multiple times (Trac #166) - Fixed synchronizing problem with clients publishing repeatedly (Trac #124) - RTMPT connections can be closed from the serverside (Trac #179)

Red5 0.5 (2006-07-25)

New features: - Frame dropping for live streams depending on available bandwidth - Added “receiveAudio”, “receiveVideo” and “send” for streams - Destination of recorded streams can be customized (Trac #73) - VOD stream flow control adapts bandwidth based on buffer time (Trac #63) - Up-/downstream bandwidth can be specified

Bugfixes: - Only the same instances are serialized as references (Trac #58) - Re-added JSP support in manifest file of red5.jar (Trac #59) - “tagPosition” is updated in FLVReader when seeking (Trac #55) - Automatic subsopes of the host scope are disabled so only connections to existing applications are possible - Running “ant” after setup keeps wrapper configuration (Trac #76) - MP3 files with unsupported sample rates are detected (Trac #66) - Timestamps of recorded .flv files were wrong sometimes (Trac #78) - Stream types could be reused leading to a ClassCastException (Trac #84) - “ns.pause” working if no flag given (Trac #67) - A keyframe is sent for paused streams when seeking

Red5 0.5rc1 (2006-07-11)

New features: - Refactored streaming code - Refactored scope services - Refactored rtmp message de-/encoding - Enabled subsopes - Bandwidth control for on-demand streams - Experimental support for serverside streams - Added dynamic “onMetaData” for mp3 streams - Added persistence for scopes and shared objects - Added support for simple “directory-only” applications - Added remoting client support (sync / async) - Added deserializer for RecordSet remoting results - Arbitrary objects can be registered as service handlers - IClientRegistry can be customized for each scope - WEB-INF directories are added to the classpath (Trac #27) - Clients can be rejected with a custom error message - Basic “onMetaData” is generated dynamically for .flv files without any meta data (Trac #23)

Bugfixes: - MP3 files that have their protection bit set - MP3 files encoded MPEG 2, Layer III (Trac #15) - MP3 files with incomplete last frame - Shared objects bugfixes (Trac #11, #22, #25) - Application handlers were not called on disconnect - IConnection.close() now closes connection (Trac #19) - Connecting to non-existent applications returns correct error now - Jetty correctly runs on all virtual hosts (Trac #26) - Map objects are serialized correctly - Methods could be invoked with converted parameters before invoking them with the original parameters - Support invoking methods with “null” as parameter (Trac #29) - Directories for recorded files are created if they don't exist (Trac #20) - “pause(java.lang.Object, int)” was reversed for streams (Trac #16) - Serialization of arbitrary objects uses reflect api to access fields, fixes various problems with inner classes and internal objects like IConnection / IClient - Invalid stream ids are handled in “deleteStream” (Trac #21) - Stream name prefixes and names without extensions supported (Trac #28)

Red5 0.4.1 (2006–05–01)

- MP3 audio streams
- “seek” and “pause” for on-demand streams (Trac #4)
- “Address already in use” fixed after restart (Trac #5)
- Bugfixes for shared objects (Trac #6)
- Bugfixes for videoconference sample (Trac #7)
- Connection strings without hostname supported (Trac #8)
- Flash 7 version of the videoconference sample added

Red5 0.4 (2006–04–20)

- Public server-side api
- AMF remoting
- RTMPT
- Metadata API
- Basic samples and documentation

Red5 0.3 (2006–02–21)

- Live streams
- Shared objects

Red5 0.2 (2005–10–21)

- First public release
- Video streams
- Echo service