

# **Nearly There Sidekick Manual**

---

# Nearly There Sidekick Manual

Published March 2006

Copyright © 2002-2006 Nearly There Network Technologies

---

---

---

## Table of Contents

Preface .....	vi
1. Quick Start: For the Impatient .....	1
Quick Start .....	1
Detailed Setup Instructions Available .....	1
2. Getting Around Sidekick .....	2
Requirements .....	2
Installation Overview .....	2
Installing Java 1.5.0 .....	2
Alternative Servlet Engines .....	2
How Sidekick is Distributed .....	3
Installing sidekick-2.4.0.war .....	3
The Messaging server .....	4
Logging on .....	4
Starting and Stopping Sidekick .....	5
Changing Properties .....	5
Linking to Sidekick .....	6
Http Port Usage .....	6
Customizing launch pages .....	6
Displaying your images .....	8
3. Presentation Services .....	9
Overview .....	9
Private Service .....	9
Public Service .....	9
Preview Service .....	9
Service Characteristics .....	9
Streaming Option .....	10
Changing the Video Size .....	11
Pan, Tilt and Zoom (PTZ) .....	11
Emoticons .....	12
Achieving High Video Quality .....	12
Combining Shows Together .....	13
Chat Rooms .....	14
Retaining text chat history .....	14
4. Sidekick templates .....	15
Overview .....	15
CoreTool .....	15
Tool .....	16
The clientId parameter .....	16
Displaying of a Show's Recent Frame .....	17
Customizing Show Not Available graphics .....	17
Passing arguments to Sidekick Pages .....	17
Escaping Strings .....	18
Miscellaneous .....	18
5. Restricting Access .....	19
Overview .....	19
Restricting By Referrer .....	19
Restricting By Token .....	20
Restricting By Dialer .....	20
Restricting By Timer .....	21
Abusive Users .....	21
6. Snapshots .....	23
Overview .....	23
Upload Directory .....	23

Overriding Default Behavior .....	23
7. Advanced Features .....	25
Internationalization and Localization .....	25
Enabling secure pages .....	25
Show Archival .....	26
Automatically restarting a Show on Disconnect .....	27
Client Bandwidth Conservation Feature .....	27
Transitioning from private back to public .....	27
Differentiating clients by group .....	27
Hiding Sidekick Home Page .....	28
8. Pay Per View Option .....	29
Overview .....	29
Billing Support .....	29
Reporting .....	30
Support for Different Rates .....	30
Client Balance .....	30
9. Interoperability .....	31
Introduction .....	31
Remote Actions .....	31
Token Actions .....	32
Show Actions .....	32
Session Actions .....	34
Performer Actions .....	35
Service Actions .....	36
Property Actions .....	36
Miscellaneous Actions .....	37
Computer readable log files .....	38
GetToken Request .....	39
Callout Support .....	40
RemoteGateway .....	42
10. Advanced Options .....	43
Metering Tracking .....	43
Multiple Language Option .....	43
11. Operations .....	45
Overview .....	45
Optimizing Sidekick .....	45
Monitoring .....	47
Log files .....	47

---

# Preface

Sidekick is a highly configurable videochat platform, upon which you can build and deploy full-featured sites. It is a modular system built on Java J2EE (Java 2 Enterprise Edition) technology. From a bird's eye view, Sidekick consists of the following three major components:

- **Server-side capabilities:** This system is the bulk of our system. This is the portion that drives the web site, tracks shows and sessions, manages video streams and handles administrative tasks.
- **Performer module:** This program is automatically retrieved from the server and run on a performer's machine. It allows a performer to put on video/chat shows.
- **Browser-based applet:** This component allows a visitor or member to participate in a performer's video/chat show.

Sidekick helps with two areas of your site. First, it provides ways to surface the video/chat for prospects and encourages them to join your site. Second, it allows members to participate in video/chat shows put on by your performers. What follows are the general capabilities of each of these two areas:

The following ways are provided to give prospects a sense of your site and to encourage them to join:

- You can display small recent images of each live show. Each thumbnail represents a glimpse into a live, streaming show.
- You can display a real-time video of the show. Here, the prospect sees a live show at a size and frame rate of your choosing. Usually, you would select a smaller size and slower rate so as to not give away too much content.
- You can display a video/chat session. This allows the prospect to be part of a live show. He can participate fully the text chat. In addition, the video part of the screen can be configured in one of these ways:
  - You can display one or more still images. If more than one still image is specified, they will rotate through in turn, at a rate of your choosing. You can have different images for each performer.
  - You can display a video stream from the show. You can specify the size and frame rate so as to only tease the prospect.

As for members, a member will be able to participate in a video/chat show using the full capabilities of the system. In this mode, he can view the live video and send and receive text messages at the same time. He will be identified to the performer as a member rather than a prospect, so she can correctly respond to each.

A myriad of options allow you to configure Sidekick to your unique needs. Web-based administration screens make configuring Sidekick very intuitive.

---

# Chapter 1. Quick Start: For the Impatient

## Quick Start

Here's our absolute minimum Quick Start Instructions which will bring up Sidekick on your Red Hat Linux server. Later on, be sure to peruse the other sections of this manual for advanced settings and features.

1. Extract `sidekick-2.4.0.war` from `sidekick-kit-2.4.0.tar.gz` and install in your servlet engine.
2. Expand `application-sidekick-basis-2.4.0.tar.gz` to its "application directory" -- we suggest `~/sidekick`.
3. Configure your servlet engine as follows:
  - a. to recognize the `/sidekick` web context,
  - b. to associate the Sidekick war file (`sidekick-2.4.0.war`) with the `/sidekick` web context,
  - c. and for `/sidekick` web context to recognize the environmental variable `application-Directory` as where the Sidekick application resides. (This step is not necessary if you follow our recommendations, which is to name the application directory `sidekick` in your home directory.)

(See further down in this manual for snippet samples to use.)

4. Modify `site.properties` with correct paths for the various log files.
5. Restart your servlet engine to bring up the servlet engine with Sidekick.

Now, you can go to Sidekick's home page by browsing to `http://www.yourDomain.com/sidekick`. Log on as "admin" with password "admin" and enter your valid license key from Nearly There. Log on as "albert" with password "albert" and go to the performer's lounge and host a show. Then, to view the show, just click on "Members Area." (We recommend running the performer and client on separate machines. Nevertheless, with impaired performance, you can run both on the same machine.)

## Detailed Setup Instructions Available

We have very detailed, step-by-step instructions which are readily available. Just ask us! These instructions start with a Red Hat Linux machine and walk through all the steps necessary to set up a fully functional system. Included in the set up are the installation of the following major components:

- Java 2 1.5.0 Programming Language
- Apache 2.0.58 Web Server
- Apache's Jakarta Tomcat 5.5.17 Servlet Engine
- Sidekick 2.4.0

---

# Chapter 2. Getting Around Sidekick

## Requirements

Sidekick runs on a server with the following capabilities:

- Red Hat Linux server 8.1, 9.0, or Fedora; clones such as CentOS are also supported.
- Pentium IV processor or better (The AMD equivalent is fine);
- At least 256M of memory;
- At least 1G hard disk space (More space may be required if the show archiving feature is enabled);
- Sun's Java 1.5.0; and
- Apache Jakarta Tomcat 4.1 or higher and Apache 2.0.

Sidekick's performer program runs on a computer with the following capabilities:

- Windows 98, ME, 2000, XP;
- 800 MHz Pentium III processor or better (AMD equivalent is supported, 1.5G Pentium IV processor for streaming option or for Pan-Tilt-Zoom feature);
- 128M
- 128 Kbps Internet connection or higher;
- Standard webcam device which includes Video For Windows support support;
- Microphone for audio capture; and
- Speaker (for alerts as clients enter and leave).

Sidekick's applet client that runs on a computer with the following capabilities:

- Windows 98, ME, 2000, XP;
- 450 MHz Pentium III processor or better (AMD equivalent is fine.);
- 56 Kbps Internet connection or higher;
- Speaker (for audio playback); and
- Internet Explorer 6.0 or above, Netscape 6.0 or above, or Firefox 1.0 or above. Java applet support must be enabled.

## Installation Overview

When installing Sidekick, you will only install Sidekick's server-side components. Components for the performer program and client applets are installed automatically as part of this process, so no specific attention needs to be paid to them.

## Installing Java 1.5.0

Java 1.5.0 is available from Sun Microsystems. Two variations are available, a JRE (Java run-time environment) and an SDK (Software development kit). Download and install the SDK. Even though Sidekick does not require any features from the SDK, many servlet engines require the SDK for compiling JSP pages. To download, visit <http://java.sun.com/j2se/1.5.0/download.html>.

## Alternative Servlet Engines

Sun Microsystems has published a specification for version 2.3 of their Servlet Engine. Sidekick requires an implementation that is compliant with version 2.3. We have tested our software with several

implementations and even though we recommend Apache Jakarta's Tomcat, any of the following are sure to work:

- Apache Jakarta's Tomcat (or Catalina) 5.5.17. See <http://jakarta.apache.org/tomcat/index.html>
- Caucho Technology's Resin 2.1. See <http://www.caucho.com>
- Mortbay's Jetty 4.1. See <http://mortbay.com/jetty/index.html>

## How Sidekick is Distributed

Sidekick is distributed as two tar files, under the names `sidekick-kit-2.4.0.tar.gz` and `application-sidekick-basis-2.4.0.tar.gz`. The first tar contains the Sidekick binary and the second contains file that represent our sample application, the same as what is demonstrated on the <http://www.nearlythere.net> site. Our sample application is a good point to start your customization.

Also, in `sidekick-kit-2.4.0.tar.gz`, are a few other files of interest. Review the `readme.txt` file for the most current information. In the docs directory, `SidekickManual.html` - - and `SidekickManual.pdf` -- provide the documentation that you are now reading. Of course, `sidekick-2.4.0.war` represents the Sidekick system and is run within your servlet engine.

## Installing `sidekick-2.4.0.war`

`sidekick-2.4.0.war` should be installed in your servlet engine. The details of this are specific to your servlet engine, but basically involve updating the servlet engine's configuration file to add a new web context, something like `/sidekick`. In addition, you will specify where the corresponding war file is located.

For your convenience, we have described in the next section how to install `sidekick-2.4.0.war` into Tomcat 5.5.17.

### CONFIGURING TOMCAT 5.5.17

You can add a file named `sidekick.xml` in Tomcat's configuration area. The directory you add it to should already exist and should be named `conf/Catalina/localhost`.

```
<Context path="/sidekick"
  docBase="/home/sidekick/sidekickWars/sidekick-2.4.0.war"
  debug="0" privileged="true" reloadable="false" >
</Context>
```

Sidekick needs to know where its configuration files are located. The application directory is where Sidekick will look for all application data, such as templates, properties and data. The above section assumes that Sidekick's application directory follows our recommendation and is a `sidekick` directory in the user's home directory. If this isn't the case, you can explicitly specify the desired application directory:

```
<Context path="/sidekick"
  docBase="/home/sidekick/sidekickWars/sidekick-2.4.0.war"
  debug="0" privileged="true" reloadable="false" >
  <Environment name="applicationDirectory"
    value="/home/sidekick/sidekick"
    type="java.lang.String" override="false"/>
</Context>
```

You will notice that the location of Sidekick's application directory is specified, which is `/home/sidekick/sidekick`. The first `sidekick` represents the Linux user account and the second `sidekick` is our recommended name for the application directory.

#### VERIFYING CORRECT OPERATION

In order to verify that Sidekick is operating correctly including the video/chat feature, you can do the following:

1. Make sure that the servlet engine is running with the Sidekick application. Enter `http://www.yourDomain.com/sidekick`. You should see Sidekick's main menu.
2. Login as a performer (login: "albert" password: "albert"), go to the Performer's Lounge and launch the client. If you get the login screen, that's good! If you get a server down message, check that the chat server is up and running.
3. Back at the main menu, logout, go to the "Members Area" and join a show. You should see the applet appear with the video/chat.

## The Messaging server

Our messaging server is responsible for maintaining connections with the performer and clients, as well as transferring messages, including status, chat, video and audio messages.

It is integrated into our web server software and is started automatically when Sidekick is started. One way to verify that the chat server is running is to log into Sidekick's admin area and click on Monitor. You will see a section like this:

```
Chat Server
* Up-and-running: Ok
* Port: 2224
```

Another way is to look at the screen output when starting Sidekick or by looking in the log file `logs/site.log`. You should see a line similar to:

```
Chat Server listening to port 2224
```

## Logging on

When installing Sidekick or maintaining it, you might need to periodically log onto your Linux server. When you log in, you will be presenting with a secure Linux shell. Within the shell, you can enter any commands. For example, you can enter commands to start and stop Sidekick.

The following steps allow you to log onto your server from a Windows machine:

1. Download Putty from <http://www.chiark.greenend.org.uk/~sgtatham/putty/>. You really only need to download `putty.exe`, but you can download `putty.zip` if you would like their entire suite of programs.
2. Save `putty.exe` to a directory from which you wish to run it.
3. On the Windows Desktop, click on **Start**, then **Run...** and type in something like `c:\bin\putty` depending on where you saved `putty.exe`. The PuTTY Configuration Window should appear.
  - a. Enter the Host Name (e.g., `www.yourDomain.com` or the ip address).
  - b. Change the Protocol to SSH.
  - c. Enter Saved Sessions (a memorable name such as "Sidekick Server").

- d. And click on **Save** to save these settings for next time.
4. Click on **Open** to actually make a connection to your server. The first time you make a connection, Putty will display a PuTTY Security Alert. If you choose to continue, click on Yes.
5. You will then be asked for the login and password. When these are entered, you will be in a secure shell and able to enter Linux commands such as **ls** (list directory) and **df** (report disk space usage).

## Starting and Stopping Sidekick

Sidekick is started and stopped from the Linux shell. Technically, you will be starting and stopping Tomcat, the servlet engine which Sidekick runs within. Depending upon how Sidekick and Tomcat were installed, there may be more than one way to start and stop Sidekick.

To start Sidekick:

- Enter the command **startup**. This is the easiest way. Usually, on installation, this alias is set up.
- Enter the command **~/startup.sh**. This shell script is usually installed when setting up.
- Enter the command **~/jakarta-tomcat-5.5.17/bin/startup.sh**. The path to tomcat might vary slightly depending on your installation and exact version.

To stop Sidekick:

- Enter the command **shutdown**. This is the easiest way. Usually, on installation, this alias is set up.
- Enter the command **~/shutdown.sh**. This shell script is usually installed when setting up.
- Enter the command **~/jakarta-tomcat-5.5.17/bin/shutdown.sh**. The path to tomcat might vary slightly depending on your installation and exact version.

When you are restarting Sidekick, you will first stop it, wait a little bit, and then start it. You can determine that Sidekick has completely shut down by entering

```
ps alx |grep java
```

and by observing only the `grep java` line in the output.

**Note:** This section assumes that you have installed Sidekick using our detailed Linux setup instructions (available upon request). These instructions add a user account "sidekick" and Sidekick is always run as this non-privileged user. Do NOT start Sidekick from root!

## Changing Properties

Most Sidekick properties can be changed in Sidekick's admin area. These properties are stored in `sidekick.properties`. Some infrequently changed options of Sidekick are contained within `site.properties`. Both `sidekick.properties` and `site.properties` are located in your application directory. This application directory is where all your customized files live. This `site.properties` file contains some properties that configure Sidekick including:

- Locations of log files,
- Names and passwords of administrator, and
- Some operational characteristics.

To configure Sidekick, we suggest you review the admin area and the `site.properties` file. For `site.properties`, you can read through it in order to understand each property and be able to enter the appropriate information or choice.

## Linking to Sidekick

When your site generates the member's area page that has a Sidekick link on it, you will insert a link such as:

```
<a href="http://www.yourDomain.com/sidekick/whosLivePublic.m">
    Click for video!
</a>
<a href="http://www.yourDomain.com/sidekick/whosLivePrivate.m">
    Click for video!
</a>
```

When the either link is clicked on, our “who's live” page will be displayed. The first link is meant for prospects and the second link for members. Refer to the section called “Customizing launch pages” for information on customizing the displayed pages. Also, refer to Chapter 5, *Restricting Access* for how to modify the second link so that only members are allowed to join a show.

## Http Port Usage

Sidekick uses two or three http ports for communication.

First, all standard web page traffic uses port 80. This is an Internet standard. If you have reason to change this port, you can change it in your servlet engine's configuration file. For example, with Tomcat, in `server.xml` search for `port="80"` and change it as desired.

Second, if you have secure sockets enabled, then port 443 is used. This is also an Internet standard.

And third, streaming video and chat messages are routed over port 2224. You can change this to another port by modifying the entry in `site.properties`.

If users are encountering access problems, it may be because our chat port has been blocked by a fire-wall. Selecting a different port might make sense.

## Customizing launch pages

When you enter our web pages from your site, you'll see our default web pages. These pages don't have your look and feel. This section describes how you can replace our pages with your own look and feel.

The following are the main pages that you can customize:

- `whosLivePrivate.vm`: This is the first page a member will encounter. It is our who's live page, which shows you who currently has a show, and a link to join the show.
- `whosLivePublic.vm`: This is the first page a non-member will encounter. It is our who's live page, which shows you who currently has a show, and a link to join the show.
- `launchPrivate.vm`: This page is displayed while a member is participating in a members only show.
- `launchPublic.vm`: This page is displayed while a member is participating in a non-member show.
- `publicWrapper.vm`: This is the html that non-member pages are displayed within.
- `privateWrapper.vm`: This is the html that member pages are displayed within.
- `tokenBalance.vm`: This is the html that displays a client's available balance. (This page is only used with the Pay per View option.)
- `profile.vm`: This is the html that displays details about a performer.
- `gallery.vm`: This is the html that displays photos of a performer.
- `galleryDetail.vm`: This is the html that displays a single photo of a performer.

The following are the feedback pages that you can customize:

- `feedback/noSession.vm`: This page is displayed if Sidekick is unable to reconcile this request with an on-going session. This normally means that the user has clicked on a link past the session timeout period, usually set to something like two hours.
- `feedback/showUnavailable.vm`: This page is displayed if the member has clicked to launch from the who's live page and the show has already been terminated.
- `feedback/systemError.vm`: This page is displayed if Sidekick was unable to complete the request. It indicates an abnormal, internal condition. If this screen appears, it should be analyzed in detail since it might mean that a component of Sidekick is not functioning correctly.
- `feedback/insufficientPrivileges.vm`: This page is displayed if an attempt was made to access a page without the proper credentials. This normally means that the page was accessed without supplying a valid token.
- `feedback/showBusy.vm`: This page is displayed if the client attempts to join a live show and is denied access. This will happen if member has clicked to join a private show and there is already someone who has joined as private. (occurs only if the *Restrict to a single client* field for this service is checked.) `showBusy.vm` will also be displayed if anyone tries to join after the performer has locked the show by checking the *Lock show* checkbox.
- `feedback/insufficientFunds.vm`: This page is displayed if the member has clicked to join a private show and the token doesn't have any time. (This feedback page will only display if the property *frontend.token.enable* is set to true.)
- `feedback/generalError.vm`: This page is displayed if Sidekick was unable to complete the request. It indicates a condition that should not appear under normal operation.
- `feedback/mustBeMember.vm`: This page is displayed if the balance was requested by an administrator or performer.
- `feedback/inputError.vm`: This page is displayed if the parameters passed with a request (such as **launchPrivate.m**) are incorrect or missing.
- `feedback/publicWrapper.vm`: This is the html that non-member feedback pages are displayed within.
- `feedback/privateWrapper.vm`: This is the html that member feedback pages are displayed within.

A common example follows. In the members area, you can link to the whos live page. This page will be `whosLivePrivate.vm` and will be contained within `privateWrapper.vm`. A member can join a show from this page. `launchPrivate.vm` will be displayed within `privateWrapper.vm`.

If you support allowing non-members to join a show (with no video, reduced video or still images), display `whosLivePublic.vm` which will be contained within `publicWrapper.vm`. When a non-member joins a show, `launchPublic.vm` will be displayed within `publicWrapper.vm`.

You will find copies of each of these pages in our templates directory. Sidekick expects the templates directory to be in its *applicationDirectory*, which is specified when Sidekick starts up (See Chapter 2, *Getting Around Sidekick* [3]).

For advanced usage, where you need to display the whosLive and/or launch pages within a different context, the following four variations are available: `whosLivePublic2.vm`, `launchPublic2.vm`, `whosLivePrivate2.vm` and `launchPrivate2.vm`. These templates are backed by `publicWrapper2.vm` and `privateWrapper2.vm`.

When you want to place an html link to one of these pages, don't specify the page directly. In Sidekick, you specify commands, which then invoke the correct pages.

- **whosLivePublic.m** -> `whosLivePublic.vm` (`publicWrapper.vm`)
- **launchPublic.m** -> `launchPublic.vm` (`publicWrapper.vm`)
- **whosLivePrivate.m** -> `whosLivePrivate.vm` (`privateWrapper.vm`)
- **launchPrivate.m** -> `launchPrivate.vm` (`privateWrapper.vm`)
- **whosLivePublic2.m** -> `whosLivePublic2.vm` (`publicWrapper2.vm`)

- **launchPublic2.m** -> launchPublic2.vm (publicWrapper2.vm)
- **whosLivePrivate2.m** -> whosLivePrivate2.vm (privateWrapper2.vm)
- **launchPrivate2.m** -> launchPrivate2.vm (privateWrapper2.vm)
- **profile.m** -> profile.vm (publicWrapper.vm)
- **gallery.m** -> profile.vm (publicWrapper.vm)
- **galleryDetail.m** -> profile.vm (publicWrapper.vm)
- **tokenBalance.m** -> tokenBalance.vm (no wrapper!)
- **page.m** -> page.vm (no wrapper!)
- **pagePrivate.m** -> pagePrivate.vm (privateWrapper.vm)
- **pagePublic.m** -> pagePublic.vm (publicWrapper.vm)
- **forward.m** -> forward.vm (no wrapper!)
- **forward2.m** -> forward2.vm (no wrapper!)

When you enter these links in and templates, you must use our exact syntax. For example, to link to **whosLivePublic.m**, you would something like:

```
<a href="$model.linkAnchor('whosLivePublic.m')">
    Click here
</a>
```

A number of the commands above are used only in special circumstances. Commands that end with **2**, such as **whosLivePublic2.m**, provide a second page for a varied look-and-feel. Commands with **page** allow you to display an arbitrary page within the Sidekick system. Commands with **forward** let you pass the following arguments (performerId, show, service) from the url so that they can be used on the page itself.

## Displaying your images

If you have images that you wish to place on one of the vm pages, there are two ways to do this with Sidekick. The first way links in your images from outside the /sidekick context. With the second way, Sidekick manages the images themselves. You will probably find the first way easier and it will suffice for most purposes.

With the first way, you would place your images in a public directory on your web server, such as /images. To display a image on a vm page, you would then create a tag such as:

```

```

Using the second way, do the following:

- In your applicationDirectory, copy the image file to a images directory. This directory should in your application directory, at the same level as the templates directory. (Example: ~/sidekick/images)
- Specify the image with \$model.linkImage('/images/banner.gif'), substituting the actual image file for banner.gif. For example, an image tag might look like

```

```

---

# Chapter 3. Presentation Services

## Overview

Sidekick considers the presentation that a client observes and interacts with a service. In other words, a service comprises the levels of video, chat and audio that the client observes and interacts with. For example, prospects might be able to use text chat, but see only a reduced size video with a slow frame rate and no audio.

Sidekick provides three primary kinds of services:

- Private Service: used for members or paying clients.
- Public Service: used for prospects and non-members. Includes a shadow “degraded” service, active when the show has a client using the private service.
- Preview Service: used for prospects and non-members. Includes only the video stream (no chat and no audio).

Basically, you can think of public and preview as being targeted primarily for prospects and private for members.

A show will always have a private service associated with it. In addition, it can optionally have a public service.

## Private Service

Each show has a private service, representing the presentation presented to clients.

## Public Service

Public services are displayed to clients in shows which have a private service with an associated public service.

There are two variations of public services. The first is presented to prospects when there are no users in its private area. The second, called *degraded*, is presented to prospects when there is a user in its private area, generally a degraded presentation.

## Preview Service

For prospects, you can display an applet with only a video stream -- no audio or text chat. Sidekick also has a “recent frame” feature, where the page will display a recent frame from the show. You can think of preview in a similar way, except that the frame is being automatically updated regularly.

## Service Characteristics

With Sidekick you can have one or more areas where *paying* clients enter. Each of these area is uniquely configured, perhaps with different pricing and access privileges. For example, commonly a site might have one area for exclusive one-on-one sessions and another allowing for many paying sessions at the same time, with different pricings.

There are many properties that allow you to customize each service. In the Sidekick admin area, you will be able to add and change services. For example, these properties let you specify whether a public

area will be allowed, the access restrictions that will be enforced, as well as the sense of exclusivity that you wish to foster.

The Still images section lets you specify one or more images to rotate through on a preset schedule, say changing once every five minutes. You can have images before a show starts, when it is live, during intermission and after a show ends. Most commonly, this applies to intermission and will display images that convey “be right back” to the client. If more than one image is specified, then the images will rotate based on the specified rotation, such as every 30 seconds. These images, which must be 320x240, are retrieved from the `external` directory. You can also specify images unique for each performer by putting them in the `external/{loginName}` directory where `{loginName}` is the login name of the performer. Finally, these two options can be combined, allowing you to have the rotating images appear behind a smaller video stream.

You can disable audio entirely for this service. When disabled, the audio checkbox will not appear in the performer program and clients will not see the audio applet.

To further customize the videochat applet, Sidekick provides these options:

- Hide the audience list, which shows the name of the performer and all the clients in this show.
- Hide the color palette, which allows the client to change the color of text messages.
- Hide the Close button, Zoom or Send buttons.
- Display the enter text field above or below the chat history area.

Finally, you can display the applet in landscape or portrait mode, that is with the video to the left of the chat area or above it. This selection is made by editing the launch page specifying a different width and height.

## Streaming Option

The streaming option enables a more powerful video streaming system which increases the frame rate to up to 24 frames per second. This faster frame rate provides a more realistic experience for your clients. Our real-time implementation allows the client to view the video stream with a delay of just a few seconds, allowing the performer and client to carry on interactively.

Sidekick will automatically adjust the size of the stream according to the characteristics of the connection providing the client with the best possible viewing experience that the connection delivers. For example, between the server and the client, if the client is on a slower connection, permanently or temporarily, Sidekick will reduce the data flow. This provides the client with a continual experience at a slower frame rate. When the connection improves, Sidekick will recognize this and improve the frame rate. Likewise, Sidekick will adjust to the changing characteristics of the connection between the performer and the server.

To enable this feature, in the Sidekick admin area, select **h263** for the codec and reduce the frame rate (milliseconds per frame) to 42.

We recommend 42 because Sidekick will automatically govern the data flow over each connection. If you specifically want your video stream to be at a slower rate, you can adjust this parameter. The codec parameter is set to `jpeg` for the compression that Sidekick uses by default. For the streaming option, the codec parameter should be set to `h263`.

Since you can specify whether to use the streaming option for each service, you can offer this option to only certain clients.

For performers, the minimum requirement for their machine is higher. Sidekick requires at least a 1.5G Pentium IV CPU which gives it the power to apply a sophisticated encoding algorithm to the captured video data. Our encoder algorithm is software-based so no additional hardware (such as a special capture

card) is required.

For the server, additional processing power is used but the requirements depend on how many streams you will be handling simultaneously. For a busier site, a faster CPU or a dual-CPU configuration would improve performance.

## Changing the Video Size

Sidekick supports two video sizes:

- 320 x 240
- 352 x 288

which represents the width and height of the video in pixels. This setting can be changed under Setup/General.

By default, Sidekick uses 320 x 240, but this can be changed if you like. Changing the video size allows you to present the most appropriate size to your clients, giving them the best experience possible, balancing the size against the additional bandwidth that a larger size will require.

If you change the video size, you will also need to update these settings:

- Under Setup/Services, each public and private service must have its video specified the same as the overall size or in proportion to it. For example, if the overall video is 320x240, proportional sizes are 20x15, 40x30, 160x240, etc.
- Any still images that are displayed should have their size the same as the overall size. By default, the images that come with Sidekick are 320x240. These images can be found in Sidekick's external directory.
- The preview video must be sized correctly. This video is shown to non-members on the whosLive-Public page. It can be found in Sidekick's `site.properties` file by searching for "preview". This property can be changed by editing `site.properties` and restarting Sidekick.

## Pan, Tilt and Zoom (PTZ)

With Sidekick, your clients can control the cam! Using pan, tilt and zoom buttons, they are able to adjust the cam exactly as they like. This feature is done in software by capturing a larger size video from the cam and then streaming a portion of this video to clients. Since the video is zooming in from a larger captured size, loss of clarity is greatly reduced.

For example, the performer can set her cam to capture at 640x480. For this example, let's say that Sidekick is configured to stream at 320x240. Initially, the client will see the center portion of the captured video. As he pans around or zooms in and out, the desired portion of the captured 640x480 video will be displayed. Sidekick will automatically resize the video as necessary.

To enable, this feature, in the Sidekick admin area, for the desired service, check the checkbox "Enable PTZ Support." With this enabled, two things will change:

- The performer will be able to select a larger video size. Upon clicking on the "Format" button, she'll be able to select a size larger than the standard video (usually 320x240).
- A client that enters a show will see six additional buttons that allow him to move left or right, move up or down, and zoom in or out.

The performer's video will display the stream that is being sent to the client so she will know exactly what the client is doing.

## Emoticons

Sidekick's text chat provides two features to further engage performers and clients. The first allows a client to select a color that his messages will be displayed with. The second is the use of emotives, or smiles.

When Sidekick recognizes that a user has typed an emotive, it will display this as an image to better get the user's point across. The emotives that Sidekick recognizes are:

- Happy: :) , :-), :-)) , or :-)))
- Unhappy: :( , or :((
- Wink: ;)
- Surprised: :O or :o
- Grin: :D or :))
- Indifferent: :|
- Angry: :((
- Tongue: :P or :p

## Achieving High Video Quality

Sidekick provides settings which allow you to tailor its operation to the characteristics of your system which will give your members the highest quality video quality and frame rate possible.

In general, you'll find that the Sidekick is self-adjusting. Sidekick attempts to monitor the stream and dynamically adjust to varying conditions such as a net slowdown or various connection speeds on different computers. Nevertheless, the best hints you can provide via these settings will translate to the best overall video presentation.

The frame rate parameter allows you to specify the desired frame rate for all the shows of a particular service. Sidekick will attempt to stream frames from each performer at this rate. If a performer's available bandwidth is too low, the frame rate might be lowered for this performer. If your performers will be operating on higher-speed connections, a setting of 200 (representing 5 frames per second) is attainable with standard Sidekick. If your performers will be primarily on low-speed connections, a setting of 1000 (representing 1 frame per second) may be all that is attainable.

If you have the Streaming Option, you can specify a frame rate up to 24 frames per second (42 milliseconds per frame) rather than up to 5 or so frames per second.

### JPEG PUSH

The following section only applies to Sidekick's standard codec, `jpeg` which is a jpeg push technology.

Another setting provides a hint as to how large each frame should be. The size of each frame determines the quality of the image. Setting this parameter around 4000 to 7000 should provide reasonable video quality. Don't be too generous with this number unless your performers and clients are both on high-speed connections.

```
#
# Specify how large a frame to allow a performer client
# to generate.
#
# This should be in line with expected client data rate.
# For example, at 1 frames/second on a 56k line, a client
# might be able to receive one 5,000 frame per second.
# Even if the service is set to deliver 2 frames/second,
# the client will only receive one frame/second.
```

```
#  
# Alternately, if set to 7,500, then on a 56k line, a client  
# might only receive a frame every 1 1/2 seconds.  
#  
# If you are supporting many clients on 56k modems, a value  
# like 4,000 is reasonable. If most of your clients are on  
# high-speed connections, a value like 8,000 is reasonable.  
#  
business.performer.frameSizeCeiling = 7000
```

A particular frame will take a certain amount of bandwidth -- you can think of a frame as a jpeg image and could range from 2500 bytes to 40000 depending on its visual quality. On the other hand, a performer's computer can only stream so many frames per second depending on its connection speed, so we may have to give up some visual quality for a quick frame rate.

Keep in mind that the bandwidth used depends on the quality of the webcam that the performer is using. Lower-quality webcams will result in more optical distortion resulting in larger frames and thus more bandwidth.

As a specific example, let's say a performer is on a high-speed connection (say DSL) and a member is on a dial-up connection, say at 56k. This member can receive perhaps 5,000 bytes per second, so setting *frameSizeCeiling* to 4,500 might be the best we can do. With this setting, he will receive around 1 or so frames per second. You should set *millisecondsPerFrame* to 1000 so that she streams one frame per second. If other members are on faster connections, the setting might be decreased to 500, providing two frames per second. This would allow each member to have an optimal experience.

Let's say in the above example, all members were on high-speed connections. Then, we might increase *frameSizeCeiling* to 6000 or more and a frame rate of 2 frames per second.

This discussion describes some general characteristics, but note that the settings are all interrelated: First, if the performer is capturing at too high a frame rate for a member, then the member will only be able to receive and render some of them. This governing is handled automatically by Sidekick -- having the member *lose* a frame periodically is considered acceptable. Second, if *frameSizeCeiling* is too high, meaning that a frame will take more bandwidth, this will result in frames taking longer to be received by a member, thus slowing his effective frame rate. So, we suggest you don't set the bandwidth without considering your member's connection speeds.

## Combining Shows Together

Sidekick allows you to have several cams which are tied together. All text messages are sent to all performers and clients in all of the shows. Audio will originate from only one performer and any clients in any show will hear it.

Normally, Sidekick has a notion of one cam per show. This feature surfaces what can be considered a *supershow*, perhaps thought of as a single show with multiple cams.

This feature combines all shows of a particular service. The available services can be listed in Sidekick's admin area. You can edit a service, and enable this feature, essentially combining any shows which are launched with the this service. The Audio login name field specifies from which performer the audio will originate.

The use of this feature might be as follows: You can have four or so cams all in the same studio. Shows with each of these cams are started, specifying the same service at the login screen. Clients can be presented with a whosLive page which shows a recent frame from all four cams and he can select one. The launch page can present a videochat session with one of the cams, and a recent frame from the other three can also be shown. The client can switch between them at will.

## Chat Rooms

Sidekick allows you to host chat rooms. In the traditional sense, chat rooms are where many clients can converse with each other using text messages. From a Sidekick perspective, these are shows which have *no* performer! They provide a way for your clients to interact even if no live content is being offered, encouraging greater site loyalty.

Sidekick allows you to have any number of chat rooms running. Using Sidekick's administrative area, you can add rooms at will. The chat rooms will run continuously without any attention.

Each chat room has an associated service which allows you to specify various characteristics of the room. For example, if you wish to restrict your chat room to be for members only, you can add a restriction.

## Retaining text chat history

Sidekick can display recent text chat messages in the client's applet when a client joins a show. This way he will be able to understand the current conversation flow better. This feature is enabled individually for each service. Technically, if enabled, then any clients that are in a show using this service will have their chat messages saved. Also, new clients using this service will have any saved messages displayed.

When displaying the history to clients in public or private services, it takes into account the exclusivity options. So for example, if your private service is configured to not route messages from public to private, then these messages won't be displayed to new private clients.

---

# Chapter 4. Sidekick templates

## Overview

Sidekick has several html pages which it is responsible for displaying to your clients. These pages follow into the following categories:

- WhosLive pages: For displaying a list of the shows that are live (or even which performers are not live!).
- Launch pages: This is the page with the videochat applet.
- Profile page: Can display information about each performer.
- Token balance page: With the Pay per view option, this page can display a client's remaining time.
- Miscellaneous pages: Mostly feedback screens, such as informing a client that he has insufficient funds.

These pages are all located in the `templates` directory and have a `.vm` extension. They are basically plain text files that can be displayed and easily edited. They contain standard html tags plus a few special directives that Sidekick uses to insert dynamic content, such as a performer's screen name.

## CoreTool

CoreTool is a class which the templates can access for retrieving data from Sidekick. The following is a list of calls that can be made:

- `#set ($show = $model.coreTool.getLiveShow($performer))` -- given a performer, this returns an in-progress show. Note that virtual performers have multiple shows, so this should only be used for live performers.
- `#set ($show = $model.coreTool.randomLiveShow)` -- returns one of the current live shows at random
- `#set ($performer = $model.coreTool.randomPerformer)` -- return one of the non-live performers at random
- `#set ($shows = $model.coreTool.liveShows)` -- return all private service shows for the first loop
- `#set ($shows = $model.coreTool.getLiveShows(0))` -- return all private service shows for the first loop
- `#set ($shows = $model.coreTool.getLiveShows(1))` -- return all private service shows for the second loop
- `#set ($shows = $model.coreTool.allLiveShows)` -- return all live shows
- `#set ($performers = $model.coreTool.nonLivePerformers)` -- return list of performers that don't have live shows
- `#set ($loginName = 'albert') #set ($performer = $model.coreTool.findPerformer($loginName))` -- return performer by login name
- `#set ($showState = $model.coreTool.getShowState($show))` -- the showState of passed show as a String: preShow, live, intermission, postShow
- `#set ($number = $model.coreTool.getNumberSessions($show))` -- the number of clients in this show (doesn't include any administrators in the show)
- `#set ($serviceName = 'private') #set ($number = $model.coreTool.getNumberSessions($show, serviceName))` -- the number of clients in this show for specified service (doesn't include any administrators in the show)
- `#set ($yes = $model.coreTool.showSupportsLanguage($show, $languageCode))` -- does this show support a particular a client communicating in a particular language?
- `#set ($rate = $model.coreTool.getServiceRate($service))` -- returns the rate for this service, nicely formatted
- `#set ($rate = $model.coreTool.getPerformerRate($performer, $service))` -- returns the performer's

- rate, nicely formatted, e.g., \$3.99
- #set (\$yes = \$model.coreTool.isRateOverridden(\$performer, \$service) -- is this performer's rate different than the nominal rate?
- #set (\$show = \$model.coreTool.getPrimaryPhoto(\$performer)) -- given a performer, this return the photo designated as primary. Returns null if no available photo.
- #set (\$performers = \$model.coreTool.getActivePerformers()) -- returns a collection of all the active performers
- #set (\$show = \$model.coreTool.getPreviousShow(\$show, \$loop)) -- returns the show that is listed before the passed show in the whosLive list in either loop 0 or loop 1. The loop argument can be left off and the first loop will be used.
- #set (\$show = \$model.coreTool.getNextShow(\$show, \$loop)) -- returns the show that is listed after the passed show in the whosLive list in either loop 0 or loop 1. The loop argument can be left off and the first loop will be used.

## Tool

Tool is a general-purpose class. The following is a list of calls that can be made:

- #set (\$text = \$model.tool.truncate(\$text, \$max)) -- shortens the passed text string so it isn't longer than \$max characters.
- #set (\$n = \$model.tool.randomInteger(\$limit)) -- returns a random number somewhere between 1 and \$limit, where 1 and \$limit can be returned
- #set (\$text = \$model.tool.esc(\$text)) -- returns \$text expanded so is valid html
- #set (\$text = \$model.tool.escJS(\$text)) -- returns \$text expanded so is a valid Javascript string, i.e., within quotes

## The clientId parameter

Each client that the site passes to Sidekick for a videochat session can be identified by a clientId. If the clientId is not passed, Sidekick will ask the user for a name. Passing in a clientId means that a user will not be asked for a name and it also allows the site to control precisely what name shows up for each client.

The clientId should be unique for each client. This is important since Sidekick doesn't allow two clients in the same show to have the same name.

There are two ways to pass the clientId to Sidekick. The first way is available if you are using tokens and is passed as a parameter to the GetToken request, something like this:

```
http://www.yourDomain.com/sidekick/GetToken?allowance=30&clientId=fred
```

Passing a clientId this way ensures that the user will not be able to modify it in any way (since the second way, described next, will not be available for this client).

The second way is to pass clientId to the whosLive page, which is the first meaningful page displayed for a client. When requesting this page to be displayed for a client, pass the clientId as a parameter, for example:

```
<a href="$linkAnchor('whosLivePrivate.m')?clientId=fred">
  Click here
</a>
```

The `clientId` will simply be held and passed around by Sidekick. Sidekick will not have any dependencies on the string itself or perform any processing with the `clientId`. Also, the `clientId` will be passed back by Sidekick in the Begin and End of Session callouts.

## Displaying of a Show's Recent Frame

On the Whos Live page, Sidekick will display a list of shows that are currently live. In addition, a recent image from the show can be displayed to provide a user with additional details about a show before launching it.

If this feature is enabled, Sidekick will make available a relatively recent frame from the show. The frame's maximum age will depend on the setting of the `refreshFrequency` parameter in the `site.properties` file. In other words, when the Whos Live page is displayed, the image displayed for a show will be anywhere from 0 seconds old to `refreshFrequency` old.

We suggest that you be conservative with this setting so as not to eat up unnecessary bandwidth. A setting of `60000` (60 seconds) will upload only one frame from each performer each minute. For example, even if no one is in a show, frames will be periodically uploaded to the server to satisfy this requirement.

Sidekick will automatically resize the frame to the displayed image size and cache these images for quick subsequent retrieval. Since resizing frames consumes significant CPU cycles, this is another reason why a conservative `refreshFrequency` setting is appropriate.

```
#
# Force the performer client to upload a new frame at least
# with this frequency
# - specify in milliseconds
# - this feature is used in for voyeur shows where it is desired
# to have a recent frame from a show on the who's live page even
# if there are no clients
# - also used by the server to govern the retrieval of newer frames
#
chat.show.recentFrame.enable = true
chat.show.recentFrame.refreshFrequency = 60000
```

## Customizing Show Not Available graphics

On the whos live page, clients will normally see a recent frame from a show. If there is no recent frame available, Sidekick will display a graphics that says "Please standby." For example, this might occur when a show just starts up. If you would prefer to have your own graphics, you can replace it.

To replace the graphics, place an image named `notAvailableRecentFrame.jpg` in your external directory. This image should be the size that you wish your recent frames to be displayed, the same as specified in the `chat.show.recentFrame.width` and `chat.show.recentFrame.height` properties.

## Passing arguments to Sidekick Pages

When you display Sidekick pages, such as the `whosLive`, `launch` and `tokenBalance` pages, you may wish to change what is displayed based on application-specific arguments. You are able to add your own arguments when linking to these pages. Then within the page, you can then change the display appropriately.

For example, you may wish to pass a `theme` argument to `whosLivePrivate` so that it is displayed differently to different clients. In this case, you would designate one of the three available arguments -- `arg1`,

`arg2` or `arg3`, for the theme, and construct your url as follows:

```
whosLivePrivate.m?arg1=night
```

Then, in the `whosLivePrivate.vm` page, you can have lines such as:

```
#if($model.arg1 == 'night')
  #set($color = 'blue')
#else
  #set($color = 'white')
#end
<body bgcolor="$color">
```

This approach provides a method for your application to dynamically customize the Sidekick pages.

## Escaping Strings

When you wish to display text on a page, there may be various characters that will not be displayed correctly based on syntax. Here are two examples:

- With HTML, the `<` character is used to indicate the start of a tag, such as `<img>`. In order for the `<` character to be displayed correctly it must be expanded to the `&lt;` sequence.
- If you are using the text within a javascript quoted string, then the `'` character would have to be escaped to `\'`.

We provide two macros that deal with escaping:

```
#esc(text string)
#escJS(text string)
```

which can be used for escaping strings for html and javascript quoted strings, respectively.

## Miscellaneous

The profile page can be the first page that a client views when entering the Sidekick system. It will accept one of two parameters: `id` or `name`. The `id` is used by Sidekick when a client clicks on a link on the browse page. The `name` can be used if you want to specify a particular performer by login name.

---

# Chapter 5. Restricting Access

## Overview

This chapter describes how you can restrict access to Sidekick pages and sessions to only those users whom you intend. Without these restrictions, less than scrupulous people, would be able to make the video/chat accessible to themselves and those who aren't members. This section describes a system you can enable which will ensure that only intended users and members are able to access your pages and performer shows. In addition, we discuss how to manage abusive users.

There are two places in Sidekick where you can enforce a restriction:

- The display of the launch and whosLive pages. You can configure Sidekick to allow or deny the display of each page as you like. Actually, there are eight variations of these two pages, each of which can be individually configured.
- The launching of a session and its continuance. Sidekick allows you to control a client's ability to join a live show and control how long his session should last.

Sidekick has four kinds of restrictions available:

- By Referrer. When a client is forwarded from one site to another, a http referrer field is included specifying from where he came. Sidekick can limit access to only those that have arrived from a known list.
- By Token. Tokens are magic strings that contain a notion of available time. These tokens can be associated with clients, which allows Sidekick to manage the amount of time that they are allowed to be in certain shows.
- By Dialer. Dialers are systems where access is restricted to those clients that have an ip address in a known list. Sidekick can allow or deny access based on a user's ip address.
- By Timer. Sidekick can restrict a service to terminate a client's session after a certain amount of time has elapsed

The remainder of this chapter describes these restrictions in more detail.

## Restricting By Referrer

Access to the videochat portion of the site is via a link, probably located in your members area. This section describes a restriction you can enable which will ensure that only members are able to join your performers shows, for example, only those that arrive from the /members area of your site.

When a request is made from one server to another, the request includes information which indicates from where it originated.

Sidekick will maintain a list of addresses from which clients are accepted. This list can be specified in Sidekick's admin area. The list is stored in `referrers.txt` in the application directory.

In addition to the above methods of specifying referrers, Sidekick supports specifying referrers in a file named `referrers.txt` in the application directory. If this file exists, Sidekick include consider these referrers when deciding whether to accept or reject a client. The format of this file is either a comma-delimited list, or a referrer on each line.

The *referrer* restriction should be enforced on each sidekick page to which clients will be forwarded. On these pages, you should specify the *referrer* restriction using Sidekick's Setup/Restrictions page.

This restriction will ensure that a client will only see the whosLivePrivate page if he has been referred from a known address. For each attempt to display the whosLivePrivate page, Sidekick will verify that the referrer is as specified. If not, the user will be turned away by displaying the feedback/insufficientPrivileges.vm page. If accepted, Sidekick will also tag this client as authorized so that later Sidekick requests will know that he has been referred from a valid site.

In addition, using Sidekick's admin area, you should specify *referrer* for the desired service. Doing so means that when a client wishes to launch this service, Sidekick will check that he has been referred from a valid site.

Template pages can use the parameter *\$model.referrer* if different information is to be displayed based on from where a client arrived at Sidekick.

## Restricting By Token

A token is a key that is associated with a user. This token includes the notion of how long this user is allowed in a private session and is used to limit access to certain parts of the site. Briefly, this is how it works:

- As described below, your site will be able to generate a token for a valid member. A token will look like this: *3sRvM31Ajh*. It is a unique 10-character string made up of numbers and letters.
- When your site generates a page for a particular member that has a Sidekick link on it, it should insert this token as a parameter. For example, your link might look like:

```
<a href="/sidekick/whosLivePrivate.m?token=3sRvM31Ajh">
    Click for video!
</a>
```

- When a member clicks on the link and enters Sidekick, Sidekick will validate that this is a valid token and permit the member to join a show. Also, the token might limit the user to a certain amount of private time.

Tokens are generated randomly to guarantee uniqueness. Also, tokens automatically expire after a certain length of time to further reduce the likelihood of abuse.

In order to generate a token, your site should make an http get request to `/sidekick/GetToken`. This request will return a token which can then be inserted into a web page. See the section called "GetToken Request" for further details on the GetToken request.

To have Sidekick restrict access to a launch or whosLive page, you will want specify the `token` restriction in Sidekick's admin area on the Setup/Restrictions screen for both whosLivePrivate and launchPrivate pages.

In this case, Sidekick will look for a `token` parameter on this page. It will check that it is valid and display the page if it is. In addition, Sidekick tags this user with this token, so when he later attempts to launch a session, the token can be consulted for validity.

To have Sidekick restrict the launching or running of a session, use Sidekick's admin area to specify the `token` as a restriction for a service.

## Restricting By Dialer

Sidekick supports what are commonly referred to as dialers. These systems allow a customer to pay on a usage basis using a 900 number. Sidekick will maintain a list of authorized ip addresses and allow access to only those addresses. Specifically, Sidekick will be continuously informed as to which clients are

authorized and which are not so that it can allow or deny access to clients when they request a private show.

Sidekick currently supports two systems: `www.dialxs.com` and `www.phonecredits.com`, each of which is explained below.

`www.dialxs.com` informs Sidekick whenever a client is newly authorized and whenever a client is no longer authorized via a http request. When a client requests to join a private show, he will only be allowed if his ip address is authorized. Likewise, when he is no longer authorized, his session will be terminated.

`www.phonecredits.com` maintains a file on the Sidekick server with a constantly updated list of authorized clients. Sidekick will read this file whenever a client requests to join a private show to determine if he is authorized. Likewise, when he is no longer authorized, his session will be terminated.

To have Sidekick restrict access to a launch or whosLive page, you will to specify `dialer` on the Setup/Restrictions screen of Sidekick's admin area.

To have Sidekick restrict the launching or running of a session, add `dialer` in Sidekick's admin area.

## Restricting By Timer

If you have a service with which you only wish an associated session to run for a certain amount of time, you can use the timer restriction to accomplish this. For example, let's say you want to provide access to your free area (the public service) for only a few minutes. Or let's say that for a payment, a client is allowed only so much private time.

To have Sidekick restrict the running of a session, add `timer` as a restriction in Sidekick's admin area.

There is another property, `frontend.timer.expiration` which lets you specify the number of seconds that a session will last.

## Abusive Users

Sidekick allows you to control abusive users ensuring that other users' experiences will be positive. In particular, a performer is able to kick off users at will. Depending on your site, Sidekick can be configured in one of several ways.

First, using the *When a client enters a private session, all public sessions will be terminated* field in Services, you can specify what happens when a performer kicks off a client. If checked, a client will have his session immediately terminated. If unchecked, a client will remain in the show but his messages will no longer be forwarded to the performer or other clients.

For some sites, having a client's session immediately terminated isn't very effective because he can simply rejoin the show. For these sites, allowing a client to remain in the show, that is, using a *silent kickoff*, is meant to slowly frustrate him, encouraging him to go away.

In addition, using *IP blocking* field in Setup/Chat screen, Sidekick will ban his IP address for a period of time of your choosing. This way, a client will not be able to reenter a show for the next half hour or so. Blocking IP addresses indefinitely can cause many support issues because of proxy servers -- you end up blocking access for legitimate users! Since Sidekick only blocks an address for a short period, you can control abusive users with minimal impact to legitimate users.

For advanced blocking, Sidekick will block not only the address of the client but also the several around this address. The idea is that some service providers use a bank of proxies and the client might be reconnected on a nearby proxy. For example, if that client's address is `202.22.22.50` and *Block edit* is set to `2`, then blocked addresses will be `202.22.22.48`, `202.22.22.49`, `202.22.22.50`, `202.22.22.51`, and

202.22.22.52. If you set *Block edge* to 0, only the client's address will be blocked.

---

# Chapter 6. Snapshots

## Overview

Sidekick allows your performers to upload images. Two ways are provided:

- From Sidekick's home page, after logging in, a performer can click on *Upload snapshot*. The performer will be able to enter a description and an image file, which are then uploaded.
- In the performer program, the **Snapshot** button will allow the performer to capture an image from her cam and upload this image. There is a 5-second countdown to provide time for her to get ready. These images will be the same size as that used for the webcam, normally 320x240.

In addition, Sidekick allows you to manage these snapshots. In Sidekick's admin area, you can list the snapshots and edit them. For example, when initially uploaded a photo is not available. You can make a photo available so that it will show up for clients to view.

Sidekick has three pages which display these photos:

- **profile.vm**: Shows the primary photo. A photo can be designated primary in the admin area.
- **gallery.vm**: Displays all available photos for a performer.
- **galleryDetail.vm**: Shows one photo and allows you to navigate between photos.

If you don't want your performers to be able to upload images, there is a property, *business.snapshot.enable*, that allows you to disable this feature. You can also limit the size of uploads with *frontend.fileUpload.size.max*.

## Upload Directory

By default, Sidekick places each uploaded image to a directory, such as:

```
.../sidekick/data/performers/images/albert
```

where albert is the loginName of the performer. The file name will be of the form, 0.jpeg, 1.jpeg, ...

In addition, in each directory, Sidekick creates a file, *directory.xml*, which includes details about each image file.

## Overriding Default Behavior

When the performer program uploads an image, it uses an HTTP POST multipart request. By default, this post is handled by Sidekick and Sidekick places each uploaded image in a well-defined directory, as described above. Please be aware that if you override this behaviour, Sidekick will not be able to display these images on the *profile*, *gallery* and *galleryDetail* pages since it will have no knowledge of them!

If you would like control over these files, you can override the request. In *site.properties*, the properties:

```
business.snapshot.url.override = true
```

```
business.snapshot.url = http://www.yourDomain.com/yourHandler
```

are used to specify your request handler. The request will arrive with the `loginName` and password of the performer, and a description. The form names are the same, that is, `loginName`, `password`, and `description`. The image data will be contained in a separate part. Another parameter, `source` is set to `client` if the request originates from the performer program, and to `lounge` if from Sidekick's web page.

If overriding the default POST behavior, for consistency, Sidekick will also use your url when a performer uploads images using the Sidekick web page in the performer's lounge.

---

# Chapter 7. Advanced Features

## Internationalization and Localization

Sidekick includes a number of features that allow it to adapt to your local region. In general, there are four aspects of Sidekick that can be localized:

- Performer Program.
- Administrative area.
- Client Applet.
- Overall character encoding.

Before going into detail into these three areas, we should point out that Sidekick's text chat supports many different languages. In fact, all European and asian languages are supported. The right-to-left languages, such as hebrew, are currently not supported.

The Performer Program will automatically detect the region the machine is running on and configure itself appropriately. If Sidekick includes language support for this region then all text will be displayed in this language. If not, English is displayed.

The Administrative area is always displayed in English. You can still enter information, such as screen names and other details, in the language of your choice. Note that the overall character encoding is used to enter and display the page, so it must be set appropriately.

The Client Applet will automatically detect the region for which the browser is configured. If Sidekick includes language support for this region then all text will be displayed in this language. If not, English is displayed.

Using the Setup/General menu in Sidekick's administrative area, you can set the character encoding to either *ISO-8859-1* or *UTF-8*. *ISO-8859-1* is a character set that includes English and most european languages and is suitable for these parts of the world. For asian languages, use *UTF-8*. The following areas are effected by the character encoding:

- Template files, the .vm files. These files will be read in using the specified character encoding. In addition, the resulting page will be sent to the browser with this encoding.
- Administrative pages. These pages will be displayed using the specified character encoding.
- RemoteAccess. When RemoteAccess.m requests are made from an outside system, the parameters are expected to be in this encoding. Also, the returned data will be in this encoding.
- GetToken. When GetToken requests are made from an outside system, the parameters are expected to be in this encoding. Also, the returned data will be in this encoding.
- Event callouts. When Sidekick calls out to an outside system, this encoding will be used.

To configure Java for a specific language, it depends on the vesion of Java. For Java 1.4.2, you would look in the `jre/lib` directory and copy the desired one, such as `font.properties.zh`, to `font.properties`. For Java 1.5, you don't have to do anything.

To configure Windows for a specific language, bring up the Control Panel and then click on Regions and Languages.

To configure your browser for a specific language, select the Options menu choice and select the desired language.

## Enabling secure pages

Sidekick allows you to have certain parts of the web site displayed using the https protocol instead of the normal non-secure http protocol. These parts include the login screen and the administrative screens since these contain sensitive information.

**Note:** This feature is provided, but we would expect it to be utilized in only the most sensitive of environments.

The first thing to do is check that your server has activated secure sockets. This involves enabling a section in your servlet engine's xml configuration file. For example, with Resin, in `resin.conf`, the entry looks like:

```
<http port='443'>
  <ssl>true</ssl>
  <key-store-type>jks</key-store-type>
  <key-store-file>
    /home/sidekick/sidekick/keystore/private.keystore
  </key-store-file>
  <key-store-password>whatever</key-store-password>
</http>
```

and for Tomcat, in `server.xml`, the entry looks like:

```
<Connector className="org.apache.catalina.connector.http.HttpConnector"
  port="443" minProcessors="5" maxProcessors="75"
  enableLookups="true"
  acceptCount="10" debug="0" scheme="https" secure="true">
  <Factory className="org.apache.catalina.net.SSLServerSocketFactory"
    keystoreFile="/home/sidekick/keystore/private.keystore"
    keystorePass="whatever"
    clientAuth="false" protocol="TLS"/>
</Connector>
```

If you are using Apache as your web server, Apache will be configured with a module called `mod_jk` to communication with Tomcat. In this case, the lines will look like:

```
<Connector className="org.apache.jk.tomcat4.Ajp13Connector"
  port="8009" minProcessors="5" maxProcessors="75"
  acceptCount="10" debug="0"/>
</Connector>
```

And second, in Sidekick's admin area, configure Sidekick by checking the **Construct pages and links for secure sockets** checkbox.

## Show Archival

Sidekick can spool a show's chat and video to disk for later administrative review. This feature is intended to provide the administrator with the capability to review a performer's interactions, perhaps for training purposes.

If this feature is enabled, key events about a show are written to disk, including all chat messages and their origin; and when clients joined or left the show. In addition, video frames are written to disk.

Sidekick provides a web-based viewing capability. Simply log as an administrator, and click on *Reports/Show Archives*.

To enable this feature, go to Setup/Chat and check the Enable archives checkbox.

The Archives Capture Frequency field is set to how often frames should be saved. Since saving frames can consume considerable disk space, a high value such as 30000 (30 seconds) might provide all the information you need, yet consume less resources. The Archives Purge field is used to limit how long archives remain before being deleted by Sidekick.

## Automatically restarting a Show on Disconnect

For some sites, it is desirable to keep a show on 24 hours a day, regardless of whether a performer is in front of the cam. In this case, if there is an Internet glitch or, perhaps the Sidekick server was restarted, then the current show will terminate. With this option, the performer program will detect this and periodically attempt to launch a new show.

See the Setup/Chat screen to change this setting.

## Client Bandwidth Conservation Feature

In the client applet, the video will stream continually. Since a user might become distracted or walk away from his computer, the streaming video may end up being unseen. Sidekick provides an option that allows the system to turn off the video if there has been no activity by the user for some period of time. Activity, in this case, means that the user has sent a chat message or clicked somewhere on the screen. If the video has been suspended by Sidekick, it will immediately restart when the user clicks on the screen or sends a text message.

## Transitioning from private back to public

When a performer is ending a private session, she might need some time to ready herself before her video is again streamed. Sidekick provides an option so that it will automatically change to Intermission when the last paying client leaves her show. Then, when she is ready, she can click on "Return to Show."

See the Setup/Chat screen to change this setting.

## Differentiating clients by group

Sidekick allows you differentiate clients by groupings that you choose. Each group will have a unique name, such as "premier" or "regular." Each client's group will be displayed to performers next to each client's name. With this information, a performer might know to focus on particular clients more than others.

You can pass a group parameter to the whosLive requests. For example, your whosLive link for a client might look like:

```
<a href="http://www.yourDomain.com/sidekick/whosLivePrivate.m?group=premier">
```

For the performer side, when this client (e.g., John) is displayed in her audience list, he will show up something like this:

```
John (premier>
```

Also, all clients of the same group will be shown in the same color to the performer.

In addition, if there is no explicit group specified, Sidekick will attempt to resolve the group by the client's ip address. In the `site.properties` file, you can specify a list of ip addresses, each with an associated group. Sidekick will compare the client's ip address to each ip address in this list (wildcards are allowed) until a match is found. If no match is found, the group remains unspecified.

## Hiding Sidekick Home Page

When you enter one of the following in your browser:

- `http://www.yourDomain.com/sidekick`
- `http://www.yourDomain.com/sidekick/home.m`

Sidekick will normally display its home page, which allows an administrator or performer to log in and work with Sidekick. Some Sidekick installations want to suppress this page in order to eliminate Sidekick branding from their site -- even though a client won't see this page unless they modify the URL in their browser.

Using Sidekick's admin screens, you can check the Hide Sidekick home page checkbox. Then Sidekick will display a simple form asking for a login and a password (instead of the usual home page).

In addition, if you feel that you would like to customize this simple form, you can create a file named `login.vm` in the `templates` directory. To understand the required html tags for this new page, use **View Source** in your browser on the login/password page that Sidekick includes.

---

# Chapter 8. Pay Per View Option

## Overview

Sidekick can be deployed to handle the video of a pay per view or per per minute web site. Sidekick surfaces features in a number of areas so that it can be utilized in a pay per view environment. With this option, you can easily build out a pay per view web site by designing your web site and simply calling into Sidekick.

Here are the key components that Sidekick adds to make this possible:

- Sidekick will allow you to track and limit the amount of time a member can spend with a performer.
- When a member runs out of time, he will be directed to a page of your choosing.
- At the end of a session, Sidekick will inform you of how much time was used by this member for your records.
- For an administrator, Sidekick will report on how much time each performer spent with members, allowing you to divide up the revenue as you see fit. Performers can also see a report of their activity.
- Chat rooms can be open to paying and non-paying members. Non-paying members can have a different video and audio presentation. In fact, when a paying member enters a show, any non-paying members can have a different presentation, perhaps a slower and smaller video or none at all.
- Sidekick can be configured so that a performer can have only one paying member at a time. It can also be configured to allow a sense of exclusivity, restricting chat messages, audio and what is displayed in the audience list.
- Performers see a list of who has joined their room. Next to each client, will be the amount of time that they have purchased and have available. This allows performers to focus on the clients that are most likely to spent their money.

## Billing Support

Sidekick is able to integrate with commonly used billing services such as proBilling, ibill, ccbill and others. We have sample scripts available. Please inquire for which services sample scripts are currently available.

Your site will ask a user which purchase option is desired, such as 10 minutes for \$15, 15 minutes for \$20, etc, and will contact a billing service for approval. Upon approval, you will request from Sidekick a token using GetToken. See the Chapter 8, *Pay Per View Option* for further details.

Sidekick will return a token, such as "FPO2IHep7i". This token is then passed when the client is forwarded for videochat to Sidekick, with a link such as "whosLivePrivate.m?token=FPO2IHep7i".

Sidekick will allow the user to join any show or shows that he wishes, as long as there is remaining time. Should he be in a show when time runs out, he will be kicked out, and sent to a specific url. This url is specified as a <param> to <applet> in the `launchPrivate.vm` page, for example:

```
#set($url = "/place_quickBuy_page_here.html")
<param name="quickBuyURL" value="$model.encodeUrl($url)">
```

Sidekick will also record the sessions the client had, which allows an administrator to generate timesheet reports. The administrator can use the report to divide up the revenue amongst performers. The performer can view her activity as well.

## Reporting

Sidekick tracks how much time each paying customer spends with each performer. As an administrator, you can view a report, over a date range, of a list of performers, each with their total time and a percentage of the grand total. This report allows you to proportion earnings according to activity.

In addition, a performer can view a report of her activity over a date range.

## Support for Different Rates

Sidekick allows each performer to have a different rate. Sidekick will manage the time that a client has with each performer. For example, if a client is with a higher priced performer, his time will run out sooner.

In order to utilize this feature, go to Sidekick's administrative section, and click on "Display rates." The nominal rate specifies the base rate. This would commonly be set to the typical rate that is charged on the site. You can change the nominal rate as desired; as well, for each performer, you can override their rate.

When a token is requested using `GetToken`, the requested time can be thought of as based on the nominal rate. Sidekick will allow a client this much time if he is with performers that charge the nominal rate. If he is with a performer at a higher or lower rate, his time will run out that much quicker or slower, respectively.

The administrator's "Timesheet" report breaks out each performer's time and revenue by the rate that it was set. This way, you'll be able to pay each performer their fair share, based on their rate.

To enhance the `whosLive` page, each show can have displayed its associated performer's rate, such as "\$4.99/minute."

## Client Balance

Sidekick maintains information about each active token including the amount of private time remaining. In order to display a page with these details, Sidekick provides a command, `tokenBalance.m`, which will display the balance using the template `tokenBalance.vm`.

Normally, the balance is shown using the token associated with the current session. However, if a token parameter is passed, then the balance if the requested token is displayed.

---

# Chapter 9. Interoperability

## Introduction

Sidekick provides several methods to allow you to completely control it externally. Let's say you wish to build a videochat web site using Sidekick. You'll want to inform Sidekick of key events, such as when a new performer is available. You'll also want Sidekick to inform you of other key events, such as when a session is starting and ending.

This chapter describes how you can communicate with Sidekick from another system. And not only this, but how you can do it securely so that your site will not be compromised.

## Remote Actions

You can control Sidekick by making http requests to `/sidekick/RemoteAccess.m`. Using Remote Access, you will be able to retrieve Sidekick's internal data as well as add and change it.

**Note:** By default, RemoteAccess is disabled for security reasons, meaning that it will not respond to any requests. To enable RemoteAccess, use Sidekick's admin area. In the Restrictions screen, you'll be able to enable RemoteAccess and turn on one or two kinds of protection, one by originating ip address and the other by a secret pass. That is, you will be able to restrict Sidekick so that the RemoteAccess request is only allowed from certain IP addresses and/or that a *secret pass* -- known only to Sidekick and the server making the RemoteAccess request -- must accompany the request.

In general, you will make an http GET or POST request. An *action* is required and specifies what the request is for. Each action has other parameters that may be required as well.

Sidekick can expose a url, RemoteAccess, which can be called as a standard http request. In this way, a trusted outside program can effect Sidekick. The url to be called will have the following format:

```
http://www.yourDomain.com/sidekick/RemoteAccess.m?action=theAction
```

where "www.yourDomain.com" is the domain that Sidekick is running on; "sidekick" is the directory path to which Sidekick is installed; and "theAction" is one of the defined actions.

Upon return, the http status code will always be 200, indicating that Sidekick has processed the request. In the body, the text will be formatted as a series of lines, with two lines required, and any others depending on the action. For example,

```
status=ok
message=Ok
```

or

```
status=error
message=Could not find performer
```

If an action is not supported, it will return

```
status=error
message=Unknown action
```

Having an external access point opens up a vulnerability to abuse. If you are not using this feature, leave it disabled. Use Sidekick's admin area to enable RemoteAccess and to restrict access.

## Token Actions

To retrieve details about a token, call the RemoteAccess Url with the following parameter:

```
action=tokenBalance&token=A1B2C3D4E5
```

To remove a token, call the RemoteAccess Url with the following parameter:

```
action=removeToken&token=A1B2C3D4E5
```

Sidekick will remove the token immediately. In fact, if there is a current session using this token, it will also be immediately terminated. If you pass a token that is not found, this call will still return success.

To remove all tokens, call the RemoteAccess Url with the following parameter:

```
action=removeAllTokens
```

Sidekick will remove all tokens immediately. In fact, if there is a current session using this token, it will also be immediately terminated.

An optional parameter, *reason* can be passed to `removeToken` or `removeAllTokens`. This parameter will be displayed to a client as the reason their session is being terminated.

To add time to a token, call the RemoteAccess Url with the following parameters:

```
action=addTimeToToken&token=A1B2C3D4E5&allowance=20
```

where *allowance* is specified in seconds. Sidekick add the requested amount of time to the token. If a session is currently using this token, it will honor the new time.

## Show Actions

To determine if a performer has a live show or not, call the RemoteAccess Url in either of the following ways:

```
action=aboutShow&loginName=albert
```

or

```
action=aboutShow&showId=int-143
```

where `loginName` is the login name of the performer and `showId` is the id of a show or chatroom. Sidekick will return a text page something like this:

```
version=2.4.0
status=ok
message=Ok
live=true
showState=live
description=Name of show
service=nude
numberSessions.private=1
numberSessions.public=6
```

The parameter `live` indicates whether the performer currently has a live show. If the show is live, another parameter, `showState`, will indicate the state: start, live, intermission or end. Another parameter, `service`, indicates the service associated with this show. Also, parameters starting with `numberSessions` will tell you how many clients are in each service. The property, `numberSessions.public`, is only returned if there is a public service associated with this show.

The parameter `description` is enclosed by double-quote characters and special encoding is done so that multiple lines can be handled, e.g., `\` becomes `\\`, `"` becomes `\"`, line feed becomes `\n`.

To list all the live shows, call the RemoteAccess Url with the following parameters:

```
action=listShows&type=show
```

where `type` is either `show` or `chatRoom`.

To send a message to all live shows, call the RemoteAccess Url with the following parameters:

```
action=broadcast&message=Smile, you're on camera!
```

where `message` is the text message to be sent. It will be displayed to each performer in a different color.

For more flexibility, the following parameters can also be passed to the broadcast action:

- `fromScreenName` - to specify the name of the sender (defaults to Administrator)
- `showId` - limits messages sent to only this show (defaults to all shows)
- `includePerformers` - sends messages to performers of live shows
- `includePrivateClients` - sends messages to clients in a private service of live shows
- `includePublicClients` - sends messages to clients in a public service of live shows

To announce to the performer something about her show, for example, that it is currently positioned on the site's home page, call the RemoteAccess Url with the following parameters:

```
action=announce&showId=int-2&title=On Home Page
```

where `title` is the text to be displayed. It will be displayed above the video, i.e., where the "No paying" text is displayed. If you wish to revert back to the default text, call with the `title` parameter as empty.

To terminate a particular show, call the RemoteAccess Url with the following parameters:

```
action=terminateShow&showId=int-21
```

where `showId` is a unique id for this show. It is unique only for this run of Sidekick and thus shouldn't be stored permanently. You can obtain the `showId` from the Begin Show callback.

To terminate a particular session, call the RemoteAccess Url with the following parameters:

```
action=terminateSession&sessionId=int-38
```

where `sessionId` is a unique id for this session. It is unique only for this run of Sidekick and thus shouldn't be stored permanently. You can obtain the `sessionId` from the Begin Session callback.

An optional parameter, *reason* can be passed to `terminateShow` or `terminateSession`. This parameter will be displayed to a performer or client as the reason their show or session is being terminated, resp.

## Session Actions

Normally, Sidekick will launch a videochat session itself using a page such as `launchPrivate.vm`. If you wish an external system to generate this page and include all the necessary tags for the videochat session, you can do that with the `getLaunchParameters` action. You would make a request as follows:

```
action=getLaunchParameters&remoteAddress=63.1.2.3&service=private&showId=int-2&t
```

*remoteAddress* should be the client's IP address. *service* should be *public* or one of the private services that are defined in Sidekick. *showId* specifies which show the client wishes to join. Optionally, you can also include the *group*, the *type*, the *clientId* and the *locale*. The *locale* parameter should be a two-letter code representing the language of the client, from the `Accept-Language` header, so that the applet will be presented in the client's local language, instead of English.

The optional parameter, *type*, can be passed to specify the kind of user. You can pass *client*, which is the default; *moderator*, to specify that the user has special privileges; or *administrator*, to specify that the user should be invisible to performers and clients.

The optional parameter, *adjustments*, can be passed to specify additional settings. It defaults to *0*. You can pass *1*, which turns on preview only public sessions. Using this setting, you can display a video only stream for public sessions. They won't be able to chat and their names won't be displayed in other clients' audience lists. The performer will be shown how many preview only clients are in her show.

Sidekick will create a notion of session for this client and return something like the following:

```
version=version=2.0.2
status=ok
message=Ok
audioArchiveAttribute=audioobs-2.0.2.jar
audioCodeAttribute=Audio
isAudioActive=false
archiveAttribute=obsobs-2.0.2.jar
isPublicService=false
chatRoomName=int-2
port=2224
language=en
uname=Fred
isChatServerIdentifyOverride=false
```

```
webapp=/sidekick-basis
options=110111111
widthVideo=320
codebaseAttribute=/sidekick-basis/clientApplet/
ticketId=int-3
chatName=localhost
heightVideo=240
codeAttribute=ATCli
token=PKsNk5T2qu
```

If you refer to the `launchPrivate.vm` page as a guide, you'll see that these parameters map directly in name and it should be straightforward to create the appropriate html code populated with this information. The information allows you to start up a videochat session with an accompanying audio component.

Commonly, an application will need to request a new token, with `GetToken`, and then request `getLaunchParameters`. In order to reduce the overhead, `getLaunchParameters` can be called without the `token` parameter and Sidekick will create a token automatically. Any parameters that are acceptable to `GetToken` can be passed to `getLaunchParameters`.

Note: If you use this technique, you will be bypassing certain checks that Sidekick makes. For example, Sidekick verifies that only authorized users can access the `launchPrivate` page. These checks would have to be handled by your external system.

## Performer Actions

To list out the login names of all administrators, performers or moderators, call the `RemoteAccess` url as follows, respectively:

```
action=listUsers&type=administrators
action=listUsers&type=performers
action=listUsers&type=moderators
```

In addition, to list out the login names of all administrators and moderators, call the `RemoteAccess` url as follows:

```
action=listPerformers&type=nonperformers
```

To retrieve parameters for a performer, call the `RemoteAccess` url with the following parameter:

```
action=getPerformer&loginName=freddy
```

In addition to the standard performer fields, a `showId` field is also returned. If it is empty, this performer does not currently have a live show. Otherwise, it includes a unique identifier for this show.

To add a performer, call the `RemoteAccess` url with the following parameters:

```
action=addPerformer&loginName=freddy&password=boy
```

Optional arguments are:

```
screenName      (default is the same as the loginName)
available       (default is true)
description     (default is an empty string)
likes          (default is an empty string)
dislikes       (default is an empty string)
custom1        (default is an empty string)
custom2        (default is an empty string)
custom3        (default is an empty string)
image          (default is an empty string)
schedule       (default is an empty string)
check1         (default is false)
check2         (default is false)
check3         (default is false)
chatLanguages  (default is any language)
rateFactor.private (default is 1.0)
```

If the performer already exists, this would be considered an error.

To update a performer, call the RemoteAccess Url with the following parameters:

```
action=updatePerformer&loginName=freddy&...
```

and follow by any parameters you wish to change. Any parameter not listed will remain unchanged. This action will not change the loginName.

The parameter *available* can be passed as true or false to make a performer unavailable -- similar to removed. This action doesn't actually remove the performer, it simply makes her unavailable. If the performer doesn't exist, this would be considered an error.

To remove a performer, call the RemoteAccess Url with the following parameters:

```
action=removePerformer&loginName=freddy
```

## Service Actions

Sidekick can return a list of all its services, call the RemoteAccess Url with the following parameter:

```
action=listServices
```

Sidekick will return a page, one service per line, with all the properties currently known to Sidekick.

Or Sidekick can return all the details about a specific service. For example, call the RemoteAccess Url with the following parameters:

```
action=getService&name=private
```

## Property Actions

Sidekick can return a list of all its properties, call the RemoteAccess Url with the following parameter:

```
action=listProperties
```

Sidekick will return a page, one property per line, with all the properties currently known to Sidekick.

Or Sidekick can return a specific property. For example, call the RemoteAccess Url with the following parameters:

```
action=getProperty&key=business.site.name
```

## Miscellaneous Actions

To ask Sidekick for its version and other relevant information, call the RemoteAccess Url with the following parameter:

```
action=about
```

Sidekick will return the following fields:

```
version=2.4.0
name=Sidekick
family=Nearly There Sidekick
javaVersion=1.5.0_06
osName=Linux
osVersion=2.4.20-19.7smp
servletVersion=Apache Tomcat/5.5.17
timezone=Pacific Standard Time
locale=English (United States)
license.key=924A0d-3CF702-4851A9-40B283
license.valid=true
license.hasPayPerView=true
license.hasMetering=true
license.hasMultipleLanguages=true
license.hasDialer=true
license.hasStreaming=true
license.expires=2005-08-11 23:59:59
characterEncoding=UTF-8
```

You can also use RemoteAccess to instruct Sidekick to verify that the callout feature, as explained in the section called “Callout Support”, is operating correctly. Sidekick will perform a "status" callout and report the results of this request. Call the RemoteAccess Url with the following parameter:

```
action=testCallout
```

Sidekick will return the following fields:

```
version=2.4.0
calloutStatus=true
calloutError=
calloutbody=Frontkick 1.0.0
```

where `calloutStatus` will be true if the `http status` field in the callout response was `Ok`; `calloutError` is an English-like description of the error if `calloutStatus` is not true; and `calloutBody` is the contents that the callout request returned.

## Computer readable log files

Sidekick can write out key events to computer-readable log files. These files can be read by an outside system, for example, to populate a database with key events.

These logs are written one event at a time, so it would probably be a periodic process, perhaps run once per day, that reads and processes them.

To enable this feature, simply enable the property in the Sidekick admin area.

The following is an example of how this file looks:

```
[Tue Jan 20 18:45:56 GMT-08:00 2004]
  COMMENT,"Initialized session logging Version 2.4.0"
[Tue Jan 20 18:45:57 GMT-08:00 2004]
  PERFORMER_LOGIN,192.168.1.14,"int-1","albert","Albert"
[Tue Jan 20 18:46:14 GMT-08:00 2004]
  CLIENT_LOGIN,192.168.1.18,"int-1","visitor","?","premier","en"
[Tue Jan 20 18:46:17 GMT-08:00 2004]
  CLIENT_CHANGE_NAME,192.168.1.18,"int-1","visitor","dan"
[Tue Jan 20 18:47:32 GMT-08:00 2004]
  CHAT,192.168.1.18,"int-1","visitor","dan","I'm dan. How are you?"
[Tue Jan 20 18:47:35 GMT-08:00 2004]
  CHAT,192.168.1.14,"int-1","albert","Albert","Fine, thanks"
[Tue Jan 20 18:47:59 GMT-08:00 2004]
  CLIENT_LOGOUT,192.168.1.18,"int-1","visitor","dan"
[Tue Jan 20 18:47:59 GMT-08:00 2004]
  PERFORMER_LOGOUT,192.168.1.14,"int-1","albert","Albert"
```

Each line starts with a timestamp of when the event occurred encoded in square brackets. Then the arguments are listed, each comma-delimited. Text strings are enclosed in double quotes, with `"` and `\` characters expanded as `\"` and `\\`, respectively.

Now each line will be explained in turn:

**COMMENT:** inserted when Sidekick starts up.

**PERFORMER\_LOGIN:** inserted when a performer starts a show. The arguments are her ip address, a unique identifier for the show, login name and screen name.

**CLIENT\_LOGIN:** inserted when a client joins a show. The arguments are his ip address, a unique identifier for the show, login name, screen name, group and language. If the screen name isn't known, `"?"` is inserted. If there isn't a group specified, `""` is inserted.

**CLIENT\_CHANGE\_NAME:** inserted when a client enters a screen name. The arguments are his ip address, a unique identifier for the show, login name and screen name.

**CHAT:** inserted when someone sends a chat text message. The arguments are the originating ip address, a unique identifier for the show, login name, screen name and the text message. Each character takes up one 8-bit character. If the original character is too big (larger than 8-bits), then the `?` character is used.

**CLIENT\_LOGOUT:** inserted when a client exits a show. The arguments are his ip address, a unique identifier for the show, login name and screen name.

PERFORMER\_LOGOUT: inserted when a performer ends a show. The arguments are her ip address, a unique identifier for the show, login name and screen name.

## GetToken Request

In order to generate a token, your site should make an http get request to `/sidekick/GetToken`. This request will return a token which can then be inserted into a web page.

**Note:** By default, GetToken is disabled for security reasons, meaning that it will not respond to any requests. To enable GetToken, use Sidekick's admin area. In the Restrictions screen, you'll be able to enable GetToken and turn on one or two kinds of protection, one by originating ip address and the other by a secret pass. That is, you will be able to restrict Sidekick so that the GetToken request is only allowed from certain IP addresses and/or that a *secret pass* -- known only to Sidekick and the server making the GetToken request -- must accompany the request.

Another variation of GetToken allows you to limit the amount of time that a client can be in private sessions. Your site will ask a user which for a purchase option, such as 10 minutes for \$15 and will contact a billing service for approval. Upon approval, you will request from Sidekick a token by making the following http request:

```
http://www.yourDomain.com/sidekick/GetToken?allowance=900
```

(allowance is specified in seconds. If not passed, defaults to 24 hours in seconds.) Sidekick will return a token, such as "FPO2IHeP7i". This token is then passed when the client is forwarded for videochat to Sidekick, with a link such as "whosLivePrivate.m?token=FPO2IHeP7i".

Sidekick will allow the user to join any show or shows that he wishes, as long as there is remaining time. Should he be in a show when time runs out, he will be kicked out, and sent to a specific url. This url is specified as a `<param>` to `<applet>` in the `launchPrivate.vm` page, for example:

```
#set($url = "/place_quickBuy_page_here.html")
<param name="quickBuyURL" value="$model.encodeUrl($url)">
```

The period that a user has to take advantage of the time is adjustable. There is a *lifespan* parameter in the properties file which can be set. Also, if you wish to change the lifespan for different tokens, something like *lifespan=240* (specified in minutes) can be added to the GetToken request. In particular, if a session is started with a token, it will be allowed to continue until either the token time has been used up, or the lifespan expires, whichever is earlier.

In addition, the *sessions* can be used to specify the number of sessions that a client will be allowed to join with this token. Commonly, this is used to limit a user to exactly one session. When this session is over, Sidekick will be able to remove the token from its memory, allowing for more efficient operation.

Further, you can also limit the duration that a client can stay in a public session. The *allowancePublic* can be specified to limit a client's session to so many seconds. If not specified, *allowancePublic* defaults to an unlimited amount of time, that is, as long as the token is valid, a public session associated with it can run indefinitely.

You can also pass the `clientId` to Sidekick. The `clientId` is the external system's notion of the client's screen name. Using GetToken, it is passed as a parameter to the GetToken request, something like this:

```
http://www.yourDomain.com/sidekick/GetToken?allowance=30&clientId=fred
```

Passing a `clientId` this way ensures that the user will not be able to modify it in any way (since the second way, where it is passed to `whosLive`, will be disabled for this client).

## Callout Support

When key events occur within Sidekick, Sidekick will call out to your site. This callout allows your site to be kept current with any activity that happens in Sidekick. For example, you can immediately record the time in seconds that a client actually was in a show.

Note: There is one special callout which is used for testing purposes only. An event "status" is passed. It is executed immediately and the result is used to respond to the "testCallout" RemoteAccess action (the section called "Miscellaneous Actions").

The following properties can be enabled to activate the callout. It is off by default for security reasons.

```
#
#
# enable http POST callback to a url when key events occur
#     begin show, change show state, end show
#     begin session, end session
#
# timestamp can be one of ntp, linux, java, standard or custom
# if "custom", then format.timestamp.pattern is used
# pattern is based on Java's SimpleDateFormat specification
#
# used to allow web site to capture key data from Sidekick
#
business.callout.enable = true
business.callout.url = http://www.yourDomain.com/sidekick/TestCalloutReceiver
business.callout.format.timestamp = ntp
#business.callout.format.timestamp.pattern = EEE MMM dd HH:mm:ss zzz YYYY
```

The following key events generate callouts:

- Sidekick has started up
- Sidekick is shutting down
- A show has begun (initiated by a performer)
- A show has changed state (initiated by a performer)
- A show has ended (initiated by a performer)
- A session has begun (initiated by a client)
- A session has ended (initiated by a client)
- Operational summary

Each callout is performed as an HTTP POST request to the address specified in `business.callout.url`. In order to ensure that this call is only made by Sidekick, the site should check the originating IP address of the request to make sure it is from the Sidekick server.

The following paragraphs describe the parameters that are sent with each event.

All callouts include:

- event (program, show or session)
- type (begin, end or changeState for show)
- timestamp

Event: *program*; Type: *begin*. Additional parameters:

- timestamp
- version (of Sidekick)

Event: *program*; Type: *end*. Additional parameters:

- timestamp
- version (of Sidekick)

Event: *show*; Type: *begin*. Additional parameters:

- showId (unique id for this show. uniqueness is guaranteed within this launch of Sidekick)
- performer (performer login name)

Event: *show*; Type: *changeState*. Additional parameters:

- showId
- performer (performer login name)
- state (of of start, intermission, live, end)

Event: *show*; Type: *end*. Additional parameters:

- showId
- performer (performer login name)
- duration (in seconds)

Event: *session*; Type: *begin*. Additional parameters:

- clientId (clientId as passed into Sidekick, empty string if not used). It is passed to Sidekick as a parameter in the whosLiveXXX.vm link.
- showId
- sessionId
- performer (performer login name)
- service (name of service)
- token
- isTokenRestricted (if tokens are actually enforced)
- ip (IP address of client)
- isAdmin (if an administrator launched this session)
- isModerator (if a moderator launched this session)

Event: *session*; Type: *end*. Additional parameters:

- clientId (clientId as passed into Sidekick, empty string if not used). It is passed to Sidekick as a parameter in the whosLiveXXX.vm link.
- showId
- sessionId
- performer (performer login name)
- duration (session length in seconds). Duration will be less than the difference between start and end times. It is based on time that the client received a quality show.
- remaining (remaining time on token in seconds)
- nominalRate (price/minute)
- service (name of service)

- token
- isTokenRestricted (if tokens are actually enforced)
- isAdmin (if an administrator launched this session)
- isModerator (if a moderator launched this session)
- ip (IP address of client)

Event: *summary*; Type: *main*. Additional parameters:

- nShows (number of live shows)
- nClients (number of clients in live shows)
- uptime (in seconds)

The timestamp parameter will be formatted in one of the following ways:

- *ntp*: Network Time Protocol, passed as hex digits, which represents a 64-bit number. The high-order 32 bits are the seconds since Jan 1, 1900; the low-order 32-bits are fractional seconds
- *linux*: integer, milliseconds since Jan 1, 1970 UTC
- *java*: integer, milliseconds since Jan 1, 1970 UTC
- *standard*: a common format which looks like "Tue Nov 05 12:33:15 PST 2003"
- *custom*: as specified by the format.timestamp.pattern property.

The pattern should be in the format that Java's SimpleDateFormat pattern expects. The pattern allows for any text format. Common components include: MM - month in year (01); MMM - month in year (Jan); dd - day in month; EEE - day in week (Tue); yyyy - year; HH - hour in day (0-23); mm - minute in hour; ss - second in minute; SSS - millisecond; zzz - Timezone (PST); and ZZZ - Timezone (-0800).

## RemoteGateway

RemoteGateway is a request that you can make to Sidekick and return desired information formatted as you like. This request will return a page that you created. To use this feature:

- Enable Remote Access in the Sidekick admin area under Setup/Restrictions.
- Create a file, `remoteGateway.vm` and place in Sidekick's templates directory.

This request accepts up to three parameters named *arg1*, *arg2* and *arg3*, which allows you to control what is returned. For example, you might pass "*arg1=getShows*" or "*arg1=isLive*" to specify what information you wish.

---

# Chapter 10. Advanced Options

## Metering Tracking

Sidekick will keep track of the bandwidth usage by referrer. In other words, any clients that arrive at the site via a referrer will have their bandwidth tracked under the referrer, providing an opportunity to charge referrers based on usage. In addition to the client's bandwidth, the performer's bandwidth is also included in the referrer's totals. The performer's bandwidth represents the video streaming from her computer to the Sidekick server plus miscellaneous small message.

Let's take a concrete example, say three clients, C1, C2 and C3 arrive at the site. C1 was referred by R1; C2 and C3 were referred by R2. R1 will include all the bandwidth used by C1 and the performer. R2 will include all the bandwidth used by C2 and C3 as well as the performer.

The totals are written out to Sidekick's `stats.log` log files on an hourly basis as well as a daily basis. For example, log lines might look like:

```
2003-05-05 14:00:00,162 [tcp-accept-80]
  INFO ChatStats - HOURLY STATS www.somewhere.com nRead 324033 nWritten 288099
2003-05-05 14:00:00,162 [tcp-accept-80]
  INFO ChatStats - DAILY STATS www.somewhere.com nRead 324033 nWritten 288099
```

The totals shown are in bytes. Bandwidth is commonly measured in bits -- multiple these numbers by eight to convert to bits. The `nRead` total refers to the total bytes received by Sidekick; and the `nWritten` total refers to the total bytes sent by Sidekick. Note that these totals represent all the chat server activity, which is a bulk of Sidekick's usage. It does not attempt to include related bandwidth, namely `whosLive` and launch web page usage and bandwidth associated with static and resized images.

By using the Linux utility, **grep**, one can readily display desired information. For example, with

```
grep DAILY chat.log | grep www.somewhere.com
```

you can review the totals for a particular referrer and add up the grand total over a desired period. A small perl program could simplify this. (Note: over time, Sidekick will be enhanced to provide an easy front-end to access this feature.)

## Multiple Language Option

The multiple language option extends Sidekick so that it can be used with clients speaking more than one language.

First, it lets administrators specify the language or languages -- up to four -- that each performer is able to communicate in. If a performer can accommodate two or more languages, she will see each language in its own chat area.

Second, Sidekick will display the `whos live` page differently depending on where a client originates. For example, French clients will only be able to choose from shows that support French. Shows from performers that can't communicate in French are not displayed.

And third, it lets the performers communicate effectively with clients. For example, if a performer speaks French and English, all French communication will appear in one area of her screen, and all English communication in another. Clients of each language are kept entirely separated.

To specify the languages that a performer can communicate in, use Sidekick's admin area to specify a performer's languages.

Languages are specified by their two-letter code. In particular these lines specify

- If the performer communicates in exactly one language and which one.
- If the performer communicates in exactly two to four languages and which ones.
- If the performer should allow clients of any language to join. If the languages property is absent, Sidekick will be interpreted this way.

As well, on the `whosLivePrivate.vm` (and `whosLivePublic.vm`) template page, you'll find lines like:

```
#if($scoreTool.showSupportsLanguage($s, 'en'))
  <a href="$model.linkAnchor('launchPrivate.m')
    &show=$model.getIdentifier($s)
    &service=Private&language=en">
    Join show
  </a>
#end
```

This `language=en` condition specifies whether a show should be displayed or not depending on whether the client supports a language or not. A `language=en` parameter can be passed to **whosLivePrivate.m** to specify the client's language. (e.g., `href="whosLivePrivate.m?language=en"`) In this case the page code might look like:

```
#if($scoreTool.showSupportsLanguage($s, 'en'))
<a href="$model.linkAnchor('launchPrivate.m')
  show=$model.getIdentifier($s)
  &service=Private&language=$model.language>
  Join Show
</a>
#end
```

The construct `"$model.language"` is replaced by the **whosLivePrivate.m**'s language parameter as passed in the url.

---

# Chapter 11. Operations

## Overview

Sidekick was designed as a zero-maintenance product. This means that on a day-to-day basis, absolutely no administrative effort should be needed to keep the system running.

For various reasons, the following sections are meant to provide a better understanding of some of the operational aspects of Sidekick. For example, these sections help answer questions such as “Is Sidekick running?”

## Optimizing Sidekick

When Sidekick operates, it can run more efficiently with some adjustments. This section reviews actions that you can take in order to optimize Sidekick's performance.

### CACHING TEMPLATES

Sidekick templates pages are read and parsed each time that a page is requested. In a live system, these pages rarely change so you can instruct Sidekick to cache these pages. This way, when a page is requested, it will already be in memory, saving the processing time of reading and parsing.

To turn on caching, change the following property:

```
velocity.file.resource.loader.cache = true
```

The following property effects how often the actual template file is checked for changes. If Sidekick notices a file has been updated, it will automatically re-read the file. Specify how often you wish to check for changes in seconds. Specifying *0* means the file will never be checked, that is, that it will always use the template that was parsed initially. This check is very inexpensive in computer time, so a small value such as *1* or *10* is recommended, which allows any occasional file changes to be recognized without having to restart Sidekick.

```
velocity.file.resource.loader.modificationCheckInterval = 10
```

### SERVING IMAGE FILES

Sidekick displays two kinds of images. The first are static images, those that don't change at all. The second are dynamic images, those that are generated by Sidekick. The recent frame feature of Sidekick, whereby a recent frame from a video stream can be displayed is an example of a dynamic image.

With respect to static images, as explained in the section called “Displaying your images”, images that you have on template pages can be stored in the web server or in Sidekick -- in the applicationDirectory area. This section pertains if you are storing your image files in Sidekick only. In this case, you can continue to benefit from the organizational advantage of Sidekick managing your image files and yet benefiting from the speed that a web server can serve up these files.

With respect to dynamic images, by enabling this feature, the following will happen with recent frames. First, recall that recent frames are generated regularly, perhaps every 30 seconds. Now, during these 30 seconds a recent frame might be requested many times by clients. Using this feature, the image will be copied into the web server space, which increases the performance by avoiding a trip to the servlet engine and the associated overhead.

The following properties enable these features:

```
frontend.webServer.documentRoot = /projects/nearlythere/resin/webapps/ROOT
frontend.webServer.images.enable = true
frontend.webServer.images.path = /sidekick!/basis/
frontend.webServer.image.enable = true
frontend.webServer.image.path = /sidekick!/dynamic/
```

*documentRoot* points to the directory that is served up by the web server with “/”. *path* is the directory path from “/” to where the files are stored. You can enable this feature for static and dynamic images individually.

In addition, you must allow Sidekick access to the web server's directory where these static and dynamic files will be stored. Sidekick needs a directory which it can write to, so from the root account, create a directory with the correct permissions with something like this:

```
mkdir /usr/local/apache-2.0.58/htdocs/sidekick!
chown sidekick:sidekick /usr/local/apache-2.0.58/htdocs/sidekick!
chatt g+w /usr/local/apache-2.0.58/htdocs/sidekick!
```

#### REDUCE LOGGING

Sidekick logs significant events to its log files, as explained in the section called “Log files”. The amount of data that Sidekick logs is controllable by logging level parameter. This parameter is normally set to *info*. Sometimes for debugging purposes, it is set to *debug*, which produces considerably more output. If you really want to reduce log output, essentially outputting only warning or error logs, the use *warn*. Ensure that the following properties are set as follows:

```
log4j.logger.net.nearlythere = info, siteLog
log4j.logger.chat = info, chatLog
```

or for even less output:

```
log4j.logger.net.nearlythere = warn, siteLog
log4j.logger.chat = warn, chatLog
```

#### SITE ARCHITECTURE

If your server isn't dedicated to Sidekick, you can also increase performance by moving other tasks off this server onto other servers. For example, if this same server is acting as a database server or your web site server, you can move these processes to their own server, thus increasing overall system performance.

#### MULTIPLE PROCESSOR SUPPORT

If your server has more than one processor (CPU), it will be utilized automatically by the following systems:

- Linux Operating System: This OS supports multiple processors internally. Thus, any process and thread management, file and socket access will run more quickly with more than one CPU.
- Apache 2.0: This web server has built-in multi-process support. Each request is handled by a separate process or thread.
- Java: Java is the underlying platform for Jakarta Tomcat and Sidekick. Wherever Tomcat and

Sidekick utilize a threading architecture, these threads can be run by different CPUs.

- Sidekick: Sidekick was designed to divide up work amongst available CPUs, so it will run more efficiently.

There is one aspect of Java that does not automatically take recognize and advantage of multiple processors. This is the garbage collection feature which is responsible for reclaiming discarded memory. In order to enable Java to divide up garbage collection amongst all available CPUs, add these options when starting Tomcat:

```
-XX:+UseParNewGC -XX:UseConcMarkSweepGC
```

In our standard deployment, you would add these options to `.bash_profile` and `startup.sh`.

## Monitoring

Sidekick provides a screen which displays a *snapshot* of the running system. To access this screen, bring up Sidekick in your browser, entering something like `http://www.yourdomain.com/sidekick`, logging in as admin (the default password is admin), and clicking on Monitor.

In the order that items appear on the Monitor page, the following are the key pieces of information:

- The exact version of Sidekick that is running. (Example: Version 2.4.0)
- How long Sidekick has been continuously running without a restart. (Example: Up Time: 17 days 4:28:13)
- What timezone, Sidekick will display dates and times in reports. (Example: Timezone: Pacific Standard Time)
- The amount of memory currently used by Sidekick. The total available memory will automatically be increased according to the run-time requirements up to a preset amount (usually set to 128M). Sidekick automatically recovers freed memory, but if you wish to encourage Sidekick, you can click on *Garbage Collect*. (Example: Used: 51,849,416 / Total: 82,472,960 (63% used))
- Is the chat server available? The chat server runs on a special port which is also displayed. If Sidekick determines that the chat server is not available, the word “Down” in red will be displayed. (Example: Up-and-running: Ok)
- The *Properties* section lets you verify that properties are correctly set.
- The *System* section displays environmental characteristics that might help technical support in identifying and resolving an issue.

## Log files

Sidekick writes significant events to log files. These log files can be very useful for ensuring an ongoing high-quality operation. For example, scanning these files periodically for errors and warnings, might allow you to proactively eliminate a problem.

If you hear of a problem with our system, you can view these files in order to understand the nature of the problem. In general, Sidekick will attempt to write out and all anomalies that occur.

Here is a description of each of the log files that Sidekick generates:

- `site.log`: this is our main log file and should be consulted first.
- `chat.log`: this stores events from the chat system.
- `stats.log`: this contains bandwidth usage for each website (referrer). This log file is only used if you have licensed the Metering Option.