

1. Documentation	2
1.1 CleanSpeak 101	2
1.2 Release Notes	2
1.3 Technical Guide	3
1.3.1 Getting Started	3
1.3.1.1 Requirements	5
1.3.2 Installation guide	5
1.3.2.1 Java	6
1.3.2.2 Database	7
1.3.2.3 Webservice	9
1.3.2.4 Management Interface	10
1.3.2.5 Search Engine	12
1.3.2.6 MySQL Connector	13
1.3.2.7 License File	13
1.3.2.8 Other JEE Web Servers	13
1.3.3 Configuration guide	16
1.3.3.1 Common Configuration	16
1.3.3.2 Webservice Configuration	16
1.3.3.3 Management Interface Configuration	17
1.3.3.4 Search Engine Configuration	17
1.3.4 Running	18
1.3.4.1 Running the Webservice	18
1.3.4.2 Running the Management Interface	19
1.3.4.3 Running the Search Engine	20
1.3.5 Scaling and Redundancy	21
1.3.5.1 Scaling the Webservice	22
1.3.5.2 Scaling the Management Interface	22
1.3.5.3 Scaling the Search Engine	22
1.3.6 WebServices	23
1.3.6.1 Content Item: Filtering Only	24
1.3.6.2 Content Item: Moderation	31
1.3.6.3 Content User	35
1.3.6.4 System User	39
1.3.6.5 Authentication	43
1.3.6.6 Standard Error Response	43
1.3.6.7 Examples	44
1.3.7 Moderation Notifications	50
1.3.7.1 Notification Request Handler	53
1.3.8 Updates and Patches	53
1.3.8.1 Upgrading from 1.x	55
1.3.8.2 Upgrading from 2.0 Pre-Releases	56
1.3.9 Troubleshooting	58
1.4 User Guide	58
1.4.1 Concepts	58
1.4.1.1 Filtering Concepts	59
1.4.1.2 Moderation Concepts	59
1.4.2 First Login	60
1.4.2.1 System Users & Permissions	61
1.4.3 List Management	61
1.4.3.1 Add/Edit Blacklist Entries	62
1.4.3.2 Testing the Filter	64
1.4.3.3 Approving Changes	65
1.4.3.4 Whitelist	65
1.4.3.5 Import/Export Lists	65
1.4.3.6 Multilingual	66
1.4.4 Moderation	67
1.4.4.1 Setup & Configuration	67
1.4.4.2 Dashboard	70
1.4.4.3 Queues	70
1.4.4.4 Escalations	70
1.4.4.5 Threaded View	70
1.4.4.6 Users & Actions	72
1.4.4.7 Content & User Search	73
1.4.5 FAQ & Troubleshooting	73

# Documentation

## Search All 2.0 Documentation

Welcome to the CleanSpeak 2.0 Documentation home. If this is your first experience with CleanSpeak, please review [CleanSpeak 101](#) for a brief guide to get started. If you're looking for documentation for another version, please visit [CleanSpeak Documentation Home](#).

## CleanSpeak 101

### Introduction to CleanSpeak

CleanSpeak is a platform for filtering and moderating user-generated content (UGC). The [Concepts](#) section of the User Guide describes how to best align these principals to your business requirements.

There are three general steps to integrating CleanSpeak with your application(s):

1. Installation
  - a. Visit the [Technical Documentation](#) for [Installation](#), [Configuration](#), [Running](#), and [Scaling](#) procedures.
2. Configuration
  - a. Use the CleanSpeak management interface to [Configure](#) the system for filtering and moderation.
3. Integration
  - a. Visit the [WebServices](#) section of the technical documentation to send content to CleanSpeak with the configuration provided by the Project Manager from the previous step.

Also, check out the [blog section](#) of our web site for frequently updated tips, tricks, and insights into community moderation and filtering.

### FAQ

How do I gain access to the software and license files?

- Login to our accounts section of the web site at <http://www.inversoft.com/login>.

Where do I get list updates and multilingual lists?

- Contact us at [support@inversoft.com](mailto:support@inversoft.com) for list(s) that you can import from the CleanSpeak management interface.

Do I need to restart CleanSpeak when installing a new license file?

- Yes, a restart is required. Note- CleanSpeak 1.x did not require a restart.

Do you have more code samples?

- Details for your specific language and using a REST API are generally available with a web search and reviewing our [WebServices Overview](#). If you'd like detailed integration services, please contact us at [support@inversoft.com](mailto:support@inversoft.com).

## Release Notes

### 2.0.1

- 2.0.1 is a bug fix release only. No database migrations or client-side changes required when updating to this version.
- Webservice
  - The webservice will now return an appropriate error when the license is invalid or has expired.
  - Improved overall error handling in all webservice api calls.
- System
  - You can now successfully delete users from the Moderators admin interface.
  - The 'Administer Users' and 'View Audit Logs' roles can now appropriately view the sections they have permission to access.
  - The Audit Log now sorts by most recent date by default.
  - Resolved an issue that would cause search engine re-indexing to sometimes corrupt content in the index.
- Filter
  - Blacklist entry definitions are now included in all export formats.
  - Added csv to available export formats.
- Moderation
  - Content Search will no longer display an unformatted error when search engines aren't available.

- Content Search form now includes an *Order by date* option to display search results in chronological order.

## 2.0

- This is a major release of the CleanSpeak platform and a number of changes have been made to the system, APIs and the database. We have worked hard to test the migration scripts and update process to ensure a smooth transition and our support team is here to help if you would like assistance. Contact us at [support@inversoft.com](mailto:support@inversoft.com) with specific questions or to set up a time for walkthrough of the entire process.

## Technical Guide

Welcome to the CleanSpeak 2.0 Technical documentation.

Search this guide:

- [Getting Started](#)
- [Installation guide](#)
- [Configuration guide](#)
- [Running](#)
- [Scaling and Redundancy](#)
- [WebServices](#)
- [Moderation Notifications](#)
- [Updates and Patches](#)
- [Troubleshooting](#)

## Getting Started

### CleanSpeak Overview

This chapter gives you a brief overview of the layout of the CleanSpeak system and what is provided to you. It also tells you things you might need to build and how the various integrations work.

This diagram gives you a general description of the possible ways that CleanSpeak is setup and integrated with your applications. As you can see near the top are various applications that you want to integrate with CleanSpeak. These applications are using the CleanSpeak Webservice for either filtering or moderation (depending on which products you need and have licensed). These applications communicate with the Webservice via a REST API over HTTP.

The Webservice loads and stores data from a relational database using SQL. This database is setup and configured by following the instructions in the Installation section of these docs. If you are using the Real-Time Filter product, the Webservice will load the white and black lists it uses to filter from the database. If you are using the Moderator product, the items you send to the Webservice to be moderated are stored in the database.

The Management Interface is the web application that is used to update the white and black lists used by the Real-Time Filter as well as to moderate content sent to the Moderator. This interface runs separately from the Webservice and communicates with the database to perform these tasks.

You'll notice that both the Management Interface and the Webservice use the same database. This allows the Management Interface to make changes to the white and black lists and then for the Webservice to query the database for those changes.

Another component in the diagram is the Search Engine. This component is used in conjunction with the Moderator product to provide full text searching of content.

The last component of this diagram is the Moderation Notification Webservice. This component is something you will need to write if you intend to use the Approval Moderation Queue features of CleanSpeak. The requirements for this component are covered in [this documentation](#).

## Next Steps

In the next chapters we will cover all of the [system and software requirements](#) for CleanSpeak.

## Requirements

### Requirements

There are a number of requirements to use CleanSpeak. There are also a number of technologies that are required depending on how you want to use CleanSpeak. Installation of all of these components (except the operating system) are covered in the [Installation chapter](#). The required technologies are:

### Operating System

CleanSpeak is capable of running on most modern operating systems. Below is a list of the supported operating systems.

- Fedora 10 or higher
- Redhat ES 5 or higher
- CentOS 5.2 or higher
- Debian 5 or higher
- Ubuntu 8 or higher
- Windows XP or higher
- Windows Server 2003 or higher
- Solaris 10 or higher
- Open Solaris 2008.05 or higher
- Mac OS X 10.5.4 or higher (64-bit Intel only)
- Mac OS X-Server 10.5.4 or higher (64-bit Intel only)

Many other operating systems are capable of running the CleanSpeak, but are not officially supported. Any operating system capable of running Java version 1.6 or higher should be capable of running CleanSpeak.

### Java

You must have Java installed to use CleanSpeak. Here are a list of the acceptable versions of Java.

- JDK 6.0 or higher from Sun
- OpenJDK 6.0 or higher
- JDK 6.0 or higher from Apple

### JEE Server for WebService and Management Interface

In order to use the Management Interface or WebService, you must have a JEE server installed. The officially supported JEE server is:

- Bundled Tomcat version 7.0.16

The CleanSpeak Management Interface and WebService are capable of running on any JEE-compliant web server. Inversoft only officially supports the bundled Tomcat releases. If you are interested in using [another JEE web server](#), Inversoft recommends Oracle's WebLogic or Sun's GlassFish. However, we do not support these web servers.

**NOTE:** Versions of Tomcat that are available for Red Hat and Debian systems via Yum or Aptitude are not valid Tomcat releases. Both of these distributions have made changes to Tomcat that are non-standard. For this reason, Inversoft does not support these versions of Tomcat.

### Database

There are a couple of different options for the database setup to use with CleanSpeak. Below are the types of databases that are currently supported.

- No database
- MySQL 5.0 or higher
- PostgreSQL 9.1 or higher

If you don't intend to use the Management Interface and want to only use the CleanSpeak Real-Time Filter using the XML database file and not a relational database then you don't need a relational database installed. If you are using the Management Interface, Moderator, or Real-Time Filter using a relational database, then you will need one of the supported relational databases installed.

### File Systems

If you are running the CleanSpeak Moderator and have content storage and searching enabled, you will need to deploy the CleanSpeak Search Engine application to a server with a fast and large enough hard-disk drive to accommodate all of your content. We suggest using a minimum of SATA 7,200 RPM drives in a RAID 1 or higher configuration.

# Installation guide

## CleanSpeak Installation

This chapter provides instructions on installing all the necessary software required to run CleanSpeak.

In most cases you will need to install the Management Interface and the WebService to use CleanSpeak. The installation instructions for those components are:

- [WebService](#)
- [Management Interface](#)

If you will be using the Moderator tool, you will also need to install the CleanSpeak Search Engine. Here are the instructions for installing that component:

- [Search Engine](#)

Regardless of which product(s) you are installing, you will need to minimally install Java and a database first. The documents that cover the installation of Java and a database are:

- [Java](#)
- [Database](#)

Due to the fact that MySQL is licensed under GPL, we are unable to ship the JDBC Connector in the CleanSpeak installation bundle. Consequently, after you have installed Java and a MySQL database, you'll need to install the MySQL JDBC Connector by hand. Instructions for doing so are provided via the link below:

- [MySQL Connector](#)

Lastly, you will need to install your license file. You can generate a license file from the [My Account](#) section of the website. Save this file and then follow the instructions in this document to install it:

- [License File](#)

## Java

### Installation – Java

The first thing you need to install to run CleanSpeak is Java 6.0 or higher. Some operating systems, such as Linux, make installing Java simple using the package management system that is part of the operating system. Other operating systems, such as Mac OS X, come with Java pre-installed. For all other operating systems you will need to download Java from Sun/Oracle here:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Java can be installed anywhere on your system as long as it is accessible and executable by the user that will be running CleanSpeak.

### Windows

Depending on the method you use to run the WebService and Management Interface, you might need to set an environment variable name **JAVA\_HOME** and set its value to the location you installed Java (usually something like c:\program files\java\jdk1.6.0\_24)

On some Windows systems there is a problem with the way that Java is installed and the files that Tomcat requires. Copy the file msvcrt71.dll from the Java bin directory (commonly located at c:\program files\java\jdk1.6.0\_24\bin) to the c:\windows\system32 directory (different versions of Windows might have different paths to the system library directory and you should consult the documentation for your version of Windows to determine where the system library directory is). If the Windows system library directory already contains the file, you do not need to copy it over.

### Debian

If you are running a Debian system, including Ubuntu, you can install Java using Aptitude by executing this command:

```
$ sudo apt-get install openjdk-6-jdk
```

### Redhat

If you are running a Red Hat system, including Fedora, Red Hat EL, or CentOS, you can install Java using Yum by executing this command:

```
$ sudo yum install java-1.6.0-openjdk
```

### **Mac OS X**

The Apple OS X operating system includes Java version 1.6.0. This version of Java is located in the directory:

```
/System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Home
```

## **Database**

### **Installation – Database**

To use the CleanSpeak Management Interface, Moderator, or Real-Time Filter using a database you will need to setup a database that will store all of the data used by CleanSpeak. Inversoft provides the necessary files for the MySQL and PostgreSQL relational databases.

First, you need to install MySQL version 5.0 or later or PostgreSQL version 9.1 or later. The database can be installed anywhere you want. Many operating systems, such as Linux, make installing MySQL or PostgreSQL simple using the package management system that is part of the operating system. Other operating systems require that you download and install one of these databases. Here are the location of MySQL and PostgreSQL:

- <http://www.postgresql.org/download/>
- <http://www.mysql.com/downloads/>

### **Debian**

If you are running a Debian system, including Ubuntu, you can install MySQL or PostgreSQL using Aptitude by executing one of these commands:

```
# For MySQL
$ sudo apt-get install mysql-server

# For PostgreSQL
$ sudo apt-get install postgresql
```

### **Redhat**

If you are running a Red Hat system, including Fedora, Red Hat EL, or CentOS, you can install MySQL or PostgreSQL using Yum by executing one of these commands:

```
# For MySQL
$ sudo yum install mysql-server mysql

# For PostgreSQL
$ sudo yum install postgresql-server postgresql
```

### **Database Creation**

If you have not already created the CleanSpeak database instance and tables, you will need to do that now. If you have already created the database, you can skip this section.

To create a new database, execute one of these commands:

```
# For MySQL
$ mysql --default-character-set=utf8 -u<root_user> -e "create database
cleanspeak character set = 'utf8' collate = 'utf8_bin';"

# For PostgreSQL
$ psql -U<root_user> -c "CREATE DATABASE cleanspeak ENCODING 'UTF-8'
LC_CTYPE 'en_US.UTF-8' LC_COLLATE 'en_US.UTF-8' TEMPLATE template0"
```

The username must be either the root user or a user that has privileges to create databases. For MySQL, this is generally a user named 'root'. For PostgreSQL, this is generally a user named 'postgres'.

You will also want to create a user that only has access to this database in order to ensure proper security. Create this user by executing these commands:

```
# For MySQL
$ mysql --default-character-set=utf8 -u<root_user> -e "grant all on
cleanspeak.* to 'cleanspeak'@'localhost' identified by '<password>'"
cleanspeak

# For PostgreSQL
$ psql -U<root_user> -c "CREATE ROLE cleanspeak WITH LOGIN PASSWORD
'<password>'; GRANT ALL PRIVILEGES ON DATABASE cleanspeak TO cleanspeak;
ALTER DATABASE cleanspeak OWNER TO cleanspeak;"
```

By default, the application is configured to connect to the database named cleanspeak on localhost with the user name cleanspeak and the password cleanspeak. For development and testing, you can use these defaults; however, we recommend a separate database server and a more secure password for production systems.

### **Table Creation**

Once you have created the database, you need to create the necessary database tables and load the initial data into the database. To create the tables you will need to download the cleanspeak-database-schema ZIP file. This contains the database script that creates the necessary tables. Once you have downloaded this file and extracted it, execute this command:

```
# For MySQL
$ mysql --default-character-set=utf8 -ucleanspeak cleanspeak < mysql.sql

# For PostgreSQL
$ psql -Ucleanspeak cleanspeak < postgresql.sql
```

### **Real-Time Filter Data**

In order for the CleanSpeak Real-Time Filter to work properly, you need to insert the initial data into the database. This data includes the profanity and other words and phrases that the filter uses. In order to load this data, you must first download the cleanspeak-database-real-time-filter ZIP file. Once you have downloaded this file and extracted it, execute this command:



```
# For MySQL
$ mysql --default-character-set=utf8 -u<username> cleanspeak <
cleanspeak-real-time-filter.sql

# For PostgreSQL
$ psql -U<username> cleanspeak < cleanspeak-real-time-filter.sql
```

## WebService

### Installation – WebService

Before you install the WebService, you must first install a Java and a database. Here are the documents that cover the installation of each:

- [Java](#)
- [Database](#)

### Bundles

There are a number of different ways to install the CleanSpeak WebService depending on the operating system and the JEE web server you will be using. Here are the various bundles and the operating system and web servers it works with:

- ZIP file
  - cleanspeak-webservice-<version>.zip
  - This bundle can be used on any operating system, but is primarily for Windows
  - Includes Tomcat version 7.0.16
- Debian package
  - cleanspeak-webservice-<version>.deb
  - This bundle can be used on any operating system that supports the Debian package management system (dpkg)
  - Includes Tomcat version 7.0.16
- RPM package
  - cleanspeak-webservice-<version>.rpm
  - This bundle can be used on any operating system that supports the Redhat Package Management system (rpm)
  - Includes Tomcat version 7.0.16
- (Experts Only) Webapp only ZIP file
  - cleanspeak-webservice-webapp-<version>.zip
  - This bundle can be used on any operating system
  - Does not include a web server and requires advanced configuration and setup. Installation of this bundle is not covered in this document

Inversoft highly recommends using the Debian, RPM, or ZIP installation bundles. The other bundles are for advanced users and system administrators that feel comfortable with Java, Tomcat, and database setup and configuration.

### Windows

To install on Windows XP or higher use the ZIP bundle. Extract the ZIP file anywhere on the file system. Remember where you extract the files. This location will be referred to as CLEAN\_SPEAK\_HOME. We suggest extracting this file to a directory such as **c:\Inversoft** on Windows.

If you want to run the WebService as a Windows Service you can install it as a Windows service using the service.bat script that ships with Tomcat. Here is how you execute that script from the command-line to install the WebService as a Windows service:

```
C:\Inversoft\cleanspeak-webservice\apache-tomcat-7.0.16\bin>service.bat
install CleanSpeakWebService
```

The last parameter to this script is the name of the service. You can use any name as long as it doesn't contain any periods ('.'), underscores ('\_') or spaces (' ').

### Redhat

To install on a Red Hat system, including Fedora or CentOS, use the RPM bundle. Execute this command to install the RPM (replace <version> with the correct version number):

```
$ sudo rpm -i cleanspeak-webservice-<version>.rpm
```

This installation process places the CleanSpeak WebService, including Tomcat 7.0.16, in the directory `/usr/local/inversoft/cleanspeak-webservice`. This location will be referred to as `CLEANSPEAK_HOME`. In addition to the files in this directory, the RPM also installs an init script named `/etc/init.d/cleanspeak-webservice`. This script will be used to start and stop the application once it has been configured.

### **Debian**

To install on a Debian system, including Ubuntu, use the DEB bundle. Execute this command to install the DEB (replace `<version>` with the correct version number):

```
$ sudo dpkg -i cleanspeak-webservice-<version>.deb
```

This installation process places the CleanSpeak WebService, including Tomcat 7.0.16, in the directory `/usr/local/inversoft/cleanspeak-webservice`. This location will be referred to `CLEANSPEAK_HOME` for the rest of this document. In addition to the files in this directory, the RPM also installs an init script named `/etc/init.d/cleanspeak-webservice`. This script will be used to start and stop the application once it has been configured.

### **Sudo**

On Linux systems, the init script uses `sudo` to run CleanSpeak as the `cleanspeak` user. Therefore, you must have `sudo` installed and must grant `sudo` privileges to the root user.

### **Next Steps**

To complete the installation of the WebService, you will need to install a license file and optionally the MySQL connector (if you are using MySQL as your database). Here are the documents for those installation steps:

- [Install the MySQL Connector](#)
- [Install your License File](#)

After you have installed Java, a database, the CleanSpeak WebService, the MySQL connector, and your license file, you are ready to configure the WebService. Here are the documents that cover the steps to configure the product:

- [Common Configuration](#)
- [WebService Configuration](#)

## **Management Interface**

### **Installation – Management Interface**

Before you install the Management Interface, you must first install a Java and a database. Here are the documents that cover the installation of each:

- [Java](#)
- [Database](#)

### **Bundles**

There are a number of different ways to install the CleanSpeak Management Interface depending on the operating system and the JEE web server you will be using. Here are the various bundles and the operating system and web servers it works with:

- ZIP file
  - `cleanspeak-management-interface-<version>.zip`
  - This bundle can be used on any operating system, but is primarily for Windows
  - Includes Tomcat version 7.0.16
- Debian package
  - `cleanspeak-management-interface-<version>.deb`
  - This bundle can be used on any operating system that supports the Debian package management system (`dpkg`)
  - Includes Tomcat version 7.0.16
- RPM package
  - `cleanspeak-management-interface-<version>.rpm`
  - This bundle can be used on any operating system that supports the Redhat Package Management system (`rpm`)
  - Includes Tomcat version 7.0.16
- (Experts Only) Webapp only ZIP file

- cleanspeak-management-interface-webapp-<version>.zip
- This bundle can be used on any operating system
- Does not include a web server and requires advanced configuration and setup. Installation of this bundle is not covered in this document

Inversoft highly recommends using the Debian, RPM, or ZIP installation bundles. The other bundles are for advanced users and system administrators that feel comfortable with Java, Tomcat, and database setup and configuration.

## **Windows**

To install on Windows XP or higher use the ZIP bundle. Extract the ZIP file anywhere on the file system. Remember where you extract the files. This location will be referred to as CLEANSPEAK\_HOME. We suggest extracting this file to a directory such as **c:\Inversoft** on Windows.

If you want to run the Management Interface as a Windows Service you can install it as a Windows service using the service.bat script that ships with Tomcat. Here is how you execute that script from the command-line to install the Management Interface as a Windows service:

```
C:\Inversoft\cleanspeak-management-interface\apache-tomcat-7.0.16\bin>service.bat install CleanSpeakManagementInterface
```

The last parameter to this script is the name of the service. You can use any name as long as it doesn't contain any periods ('.'), underscores ('\_') or spaces (' ').

## **Redhat**

To install on a Red Hat system, including Fedora or CentOS, use the RPM bundle. Execute this command to install the RPM (replace <version> with the correct version number):

```
$ sudo rpm -i cleanspeak-management-interface-<version>.rpm
```

This installation process places the CleanSpeak Management Interface, including Tomcat 7.0.16, in the directory /usr/local/inversoft/cleanspeak-management-interface. This location will be referred to as CLEANSPEAK\_HOME. In addition to the files in this directory, the RPM also installs an init script named /etc/init.d/cleanspeak-management-interface. This script will be used to start and stop the application once it has been configured.

## **Debian**

To install on a Debian system, including Ubuntu, use the DEB bundle. Execute this command to install the DEB (replace <version> with the correct version number):

```
$ sudo dpkg -i cleanspeak-filter-management-interface-<version>.deb
```

This installation process places the CleanSpeak Management Interface, including Tomcat 7.0.16, in the directory /usr/local/inversoft/cleanspeak-management-interface. This location will be referred to CLEANSPEAK\_HOME for the rest of this document. In addition to the files in this directory, the RPM also installs an init script named /etc/init.d/cleanspeak-management-interface. This script will be used to start and stop the application once it has been configured.

## **Sudo**

On Linux systems, the init script uses sudo to run CleanSpeak as the cleanspeak user. Therefore, you must have sudo installed and must grant sudo privileges to the root user.

## **Next Steps**

To complete the installation of the Management Interface, you will need to install a license file and optionally the MySQL connector (if you are using MySQL as your database). Here are the documents for those installation steps:

- [Install the MySQL Connector](#)
- [Install your License File](#)

After you have installed Java, a database, the CleanSpeak Management Interface, the MySQL connector, and your license file, you are ready to configure the Management Interface. Here are the documents that cover the steps to configure the product:

- [Common Configuration](#)

- [Management Interface Configuration](#)

## Search Engine

### Installation – Search Engine

Before you install the Search Engine, you must first install a Java. Here is the document that covers the installation of Java:

- [Java](#)

### Bundles

There are a number of different ways to install the CleanSpeak Search Engine depending on the operating system and the JEE web server you will be using. Here are the various bundles and the operating system and web servers it works with:

- ZIP file
  - cleanspeak-search-engine-<version>.zip
  - This bundle can be used on any operating system, but is primarily for Windows
  - Includes Tomcat version 7.0.16
- Debian package
  - cleanspeak-search-engine-<version>.deb
  - This bundle can be used on any operating system that supports the Debian package management system (dpkg)
  - Includes Tomcat version 7.0.16
- RPM package
  - cleanspeak-search-engine-<version>.rpm
  - This bundle can be used on any operating system that supports the Redhat Package Management system (rpm)
  - Includes Tomcat version 7.0.16
- (Experts Only) Webapp only ZIP file
  - cleanspeak-search-engine-webapp-<version>.zip
  - This bundle can be used on any operating system
  - Does not include a web server and requires advanced configuration and setup. Installation of this bundle is not covered in this document

Inversoft highly recommends using the Debian, RPM, or ZIP installation bundles. The other bundles are for advanced users and system administrators that feel comfortable with Java, Tomcat, and database setup and configuration.

### Windows

To install on Windows XP or higher use the ZIP bundle. Extract the ZIP file anywhere on the file system. Remember where you extract the files. This location will be referred to as CLEAN\_SPEAK\_HOME. We suggest extracting this file to a directory such as **c:\Inversoft** on Windows.

If you want to run the Search Engine as a Windows Service you can install it as a Windows service using the service.bat script that ships with Tomcat. Here is how you execute that script from the command-line to install the Search Engine as a Windows service:

```
C:\Inversoft\cleanspeak-search-engine\apache-tomcat-7.0.16\bin>service.bat
install CleanSpeakSearchEngine
```

The last parameter to this script is the name of the service. You can use any name as long as it doesn't contain any periods ('.'), underscores ('\_') or spaces (' ').

### Redhat

To install on a Red Hat system, including Fedora or CentOS, use the RPM bundle. Execute this command to install the RPM (replace <version> with the correct version number):

```
$ sudo rpm -i cleanspeak-search-engine-<version>.rpm
```

This installation process places the CleanSpeak Search Engine, including Tomcat 7.0.16, in the directory /usr/local/inversoft/cleanspeak-search-engine. This location will be referred to as CLEAN\_SPEAK\_HOME. In addition to the files in this directory, the RPM also installs an init script named /etc/init.d/cleanspeak-search-engine. This script will be used to start and stop the application once it has been configured.

### Debian

To install on a Debian system, including Ubuntu, use the DEB bundle. Execute this command to install the DEB (replace <version> with the

correct version number):

```
$ sudo dpkg -i cleanspeak-search-engine-<version>.deb
```

This installation process places the CleanSpeak Search Engine, including Tomcat 7.0.16, in the directory `/usr/local/inversoft/cleanspeak-search-engine`. This location will be referred to `CLEANSPEAK_HOME` for the rest of this document. In addition to the files in this directory, the RPM also installs an init script named `/etc/init.d/cleanspeak-search-engine`. This script will be used to start and stop the application once it has been configured.

## Sudo

On Linux systems, the init script uses sudo to run CleanSpeak as the cleanspeak user. Therefore, you must have sudo installed and must grant sudo privileges to the root user.

## Next Steps

After you have installed Java and the CleanSpeak Search Engine, you are ready to configure the Search Engine. Here are the documents that cover the steps to configure the product:

- [Common Configuration](#)
- [Search Engine Configuration](#)

## MySQL Connector

### Installation – MySQL Connector

In order to hook the Management Interface and/or WebService up to a MySQL database, you need to download and install the MySQL connector for Java. Unfortunately, MySQL is licensed under the GPL license, which prevents us from shipping this library inside our bundle. Therefore, you need to download the library by opening a browser and clicking this URL:

- MySQL: <http://dev.mysql.com/downloads/connector/j/>

Download the Connector/J ZIP file and extract it somewhere on your hard drive. This ZIP contains a JAR file that you need to copy to the `CLEANSPEAK_HOME/apache-tomcat-7.0.16/lib` directory.

## License File

### Installation – License File

You need to install your license file before the products will run properly. You can generate a license the from the [My Account](#) section of the website. Ensure that the license file is named `cleanspeak.license`. Copy this file into the `CLEANSPEAK_HOME/web/WEB-INF/classes` directory of the WebService and Management Interface. The Search Engine application does not require a license file.

## Updating

When your account has been updated, replace the `cleanspeak.license` file as instructed above. In order for the new license file to take effect, the WebService and Management Interface must be restarted. Refer to [Running the Webservice](#) and [Running the Management Interface](#).

Your license information displayed in the Management Interface will be updated upon restart of the service.

## Other JEE Web Servers

### Installation – Other Web Server

All of the bundles (except the webapp only bundle) come with Tomcat 7.0.16. This is the preferred web server. In some cases, you might need to run a web server other than Tomcat. If this is the case, we assume that you have a good understanding of Java and JEE. Therefore, we do not cover this installation process explicitly. However, here are some tips to get things working properly.

- The server must support JEE 1.4 or higher. If you find a server that specifies versions of the Servlet and JSP specifications here are the versions of those specifications that the server must conform to:
  - Servlet: 2.4 or higher
  - JSP: 2.0 or higher
- Ensure you have a JDBC DataSource (sometimes called a connection pool) in the global JNDI tree under one of these names
  - For the WebService - `java:comp/env/jdbc/cleanspeak-webservice`
  - For the Management Interface - `java:comp/env/jdbc/cleanspeak-management-interface`

- Ensure you have the MySQL or PostgreSQL driver JAR file installed in the necessary location for your JEE web server to properly create a global DataSource in the JNDI tree
- You might need to edit the web/WEB-INF/web.xml file in the CleanSpeak WebService or Management Interface bundle to properly configure a reference to the JDBC DataSource in the JNDI tree (this will depend on your JEE web server's requirements)
- Ensure you have a JavaMail session in the global JNDI tree under the name **java:comp/env/mail/Session**
- Ensure you have the JavaMail Framework JAR file in the necessary location for your JEE web server to create a global JavaMail session in the JNDI tree
- Configure the JEE web server to listen on an open port (generally JEE web servers listen on port 8080. You might need to use the JSVC or sudo program or run the JEE web server as a privileged user in order to bind ports below 1024)

The PostgreSQL and JavaMail JAR files listed above are located in CLEAN\_SPEAK\_HOME/lib directory. If you are using MySQL, you will need to download MySQL JAR manually and install it. [Follow these instructions to download the MySQL JAR file.](#)

### Tomcat example

As a reference to guide you in configuring other JEE web servers, here is the Tomcat 6.0.20 configuration that works with CleanSpeak:

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
  This file contains a default server configuration. Please consult the
  Tomcat
  documentation for information about modifying this file.
-->
<Server port="8010" shutdown="SHUTDOWN">

  <Listener className="org.apache.catalina.core.AprLifecycleListener"
  SSLEngine="on" />
  <Listener className="org.apache.catalina.core.JasperListener" />
  <Listener className="org.apache.catalina.mbeans.ServerLifecycleListener"
  />
  <Listener
  className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />

  <Service name="Catalina">

    <!--
      This is the main port that Tomcat listens on. If you need to change
      the port, do it
      here. You might also need to change the port specified on the root
      element of this
      XML file as well.
    -->
    <Connector port="8011" protocol="HTTP/1.1" connectionTimeout="20000"
    redirectPort="8443" />

    <!--
      Uncomment this block if you want to use Apache in front of the
      webservice
    -->
    <!--
    <Connector port="8009" enableLookups="false" redirectPort="8443"
    protocol="AJP/1.3" />
    -->

    <Engine name="Catalina" defaultHost="localhost">
      <Host name="localhost" appBase="" unpackWARs="true" autoDeploy="true"
```

```
xmlValidation="false" xmlNamespaceAware="false">
  <Context path="" docBase="../web">
    <!--
      Update this to point to your database.
    -->
    <!-- MySQL version -->
    <Resource name="jdbc/cleanspeak-management-interface"
auth="Container" type="javax.sql.DataSource"
      maxActive="20" validationQuery="select 1"
      username="cleanspeak" password="cleanspeak"
driverClassName="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost:3306/cleanspeak?relaxAutoCommit=true&character
Encoding=utf8"/>

    <!-- PostgreSQL version -->
    <!--
      Update this to point to your SMTP server.
    -->
    <Resource name="jdbc/cleanspeak-management-interface"
auth="Container" type="javax.sql.DataSource"
      maxActive="20" validationQuery="select 1"
      username="cleanspeak" password="cleanspeak"
driverClassName="org.postgresql.Driver"
      url="jdbc:postgresql://localhost:5432/cleanspeak"/>
    -->

    <!--
      Update this to point to your SMTP server.
    -->
    <Resource name="mail/Session" auth="Container"
type="javax.mail.Session" mail.smtp.host="localhost" />
  </Context>
</Host>
```

```
</Engine>
</Service>
</Server>
```

## Configuration guide

### CleanSpeak Configuration

This chapter covers the configuration required to run the CleanSpeak WebService, Management Interface and Search Engine.

In most cases you will need to configure the Management Interface and the WebService applications. Here are the instructions for configuring those application.

- [WebService](#)
- [Management Interface](#)

If you have licensed our Moderator tools, you will also need to configure the CleanSpeak Search Engine. Here are the instructions for configuring that application:

- [Search Engine](#)

### Common Configuration

#### Configuration – Common

There are a couple of common steps that you need to take to configure the CleanSpeak WebService, Management Interface and Search Engine applications. These steps are described here. The variable **CLEANSPEAK\_HOME** refers to the installation directory of all the applications.

#### Windows

In order to run CleanSpeak on a Windows server, you need to ensure that it uses the version of Tomcat that ships with CleanSpeak. Therefore, you need to remove any Tomcat environment variables that might have been previously setup on the server. These variables is generally named **CATALINA\_HOME** and **CATALINA\_BASE**.

#### All Operating Systems

In each of the applications installed in **CLEANSPEAK\_HOME**, copy the *apache-tomcat-7.0.16/conf/server.xml-example* file to *apache-tomcat-7.0.16/conf/server.xml*.

#### JVM Settings

On Linux systems that are using the init.d scripts that come with CleanSpeak to start and stop the application, if you need to override any of the JVM configuration parameters you will need to copy the *apache-tomcat-7.0.16/bin/init.d-overrides.sh-example* to *apache-tomcat-7.0.16/bin/init.d-overrides.sh* for each of the applications you have installed in **CLEANSPEAK\_HOME**. If you do not define your overrides in this file and instead use the init.d scripts, you run the risk of overwriting those changes when you upgrade to newer versions of CleanSpeak.

#### Memory Settings

Inside the *init.d-overrides.sh* file, you will find a number of different environmental variables that you can modify. If you need to change the memory settings for the JVM, uncomment the **JAVA\_OPTS** variable and modify the **-Xmx512M** setting. This controls the maximum amount of memory, in megabytes, to give the JVM. The default value is 512 Megabytes. You can set this value to anything that your server can comfortably handle. If you need assistance setting this parameter, contact Inversoft for support.

### WebService Configuration

#### Configuration – WebService

Before you configure the WebService, you must first perform the common configuration steps. Here is the document that covers those steps:

- [Common Configuration](#)

Once you have finished the common configuration, you can setup your database connection for the WebService. If you are using the WebService without a database connection and instead using the XML file, you can skip this section.



## Using a Relational Database

Open the server.xml file located at `CLEANSPEAK_HOME/cleanspeak-webservice/apache-tomcat-7.0.16/conf/server.xml` in a text editor and scroll to the bottom. If you are using PostgreSQL, comment out the MySQL Resource definition and uncomment the PostgreSQL Resource definition. Next, edit the Resource definition so that the username, password, and url attributes are correct. The url attribute needs to point to the correct server that your database is running on. Here is an example Resource element for a PostgreSQL database:

```
<Resource name="jdbc/cleanspeak-webservice" auth="Container"
          type="javax.sql.DataSource" maxActive="15"
validationQuery="select 1"
          username="cleanspeak" password="cleanspeak"
          driverClassName="org.postgresql.Driver"
          url="jdbc:postgresql://databases.example.com:5432/cleanspeak"/>
```

## Using the XML Database

If you comment out the MySQL and PostgreSQL Resource elements defined above, this will tell CleanSpeak not to use a Relational Database. Instead, it will use an XML file to load the Real-Time Filter white and black lists from. The XML file is included in the Real-Time Filter Database download located in the Downloads section of the website. Once you download that bundle, it will contain a file named **cleanspeak-real-time-filter.xml**. You will need to copy this file to the **CLEANSPEAK\_HOME/web/WEB-INF/classes** directory and rename it to **cleanspeak-database.xml**.

You can change the default location and name that CleanSpeak uses to load the XML database from. This is accomplished by copying the **CLEANSPEAK\_HOME/web/WEB-INF/config/config-production.xml-example** file to **CLEANSPEAK\_HOME/web/WEB-INF/config/config-production.xml** and editing this file in a text editor. Uncomment the **<database>** element and put the full path to the file you would like to use. This step is not required unless you are changing the location or the name of the file.

Using this type of configuration will prevent the Management Interface from sending notifications to the WebService to tell it to load new words to be filtered from the database. Instead, you will need to either edit the XML file by hand or export to XML from the Management Interface, copy the new XML file to the WebService and restart the WebService application.

## Management Interface Configuration

### Configuration – Management Interface

Before you configure the Management Interface, you must first perform the common configuration steps. Here is the document that covers those steps:

- [Common Configuration](#)

Once you have finished the common configuration, you can setup your database connection for the Management Interface.

### Datasource

Open the server.xml file located at `CLEANSPEAK_HOME/cleanspeak-management-interface/apache-tomcat-7.0.16/conf/server.xml` in a text editor and scroll to the bottom. If you are using PostgreSQL, comment out the MySQL Resource definition and uncomment the PostgreSQL Resource definition. Next, edit the Resource definition so that the username, password, and url attributes are correct. The url attribute needs to point to the correct server that your database is running on. Here is an example Resource element for a PostgreSQL database:

```
<Resource name="jdbc/cleanspeak-management-interface" auth="Container"
          type="javax.sql.DataSource" maxActive="15"
validationQuery="select 1"
          username="cleanspeak" password="cleanspeak"
          driverClassName="org.postgresql.Driver"
          url="jdbc:postgresql://databases.example.com:5432/cleanspeak"/>
```

## Search Engine Configuration

### Configuration – Search Engine

Before you configure the Search Engine, you must first perform the common configuration steps. Here is the document that covers those steps:

- [Common Configuration](#)

Once you have finished the common configuration, you can configure the location that the Search Engine will store its database files.

### **Database Location**

Open the server.xml file located at CLEANSPEAK\_HOME/cleanspeak-search-engine/apache-tomcat-7.0.16/conf/server.xml in a text editor and scroll to the bottom. There is a single configuration entry that you might want to modify. It looks like this:

```
<Environment name="solr/home" type="java.lang.String"
value="/usr/local/inversoft/cleanspeak-search-engine/data"
override="true"/>
```

This Environment entry points to a directory on the file system where the Search Engine will store all of its data. CleanSpeak defaults this to */usr/local/inversoft/cleanspeak-search-engine/data* but can be anything ranging from a local drive, NFS, SAS, or any other type of system. It is recommended that you use a local hard drive that is at least a set of 1 GB SATA 7,200 RPM drives configured as a RAID 1. We also highly recommend backing-up this directory nightly since it contains all of your indexes for searching content within the Management Interface.

Please contact Inversoft Support if you need assistance in selecting a system that will handle your volume of content.

### **Specifying Each Search Engine Server**

After you have configured the Search Engine server(s) and gotten them up and running, you will need to tell CleanSpeak where they are located. This is done inside the Management Interface under the System > Servers menu. You need to specify the URL for each Search Engine server that you have deployed.

## **Running**

### **Running CleanSpeak**

This chapter covers starting and stopping each CleanSpeak service.

- [Running the WebService](#)
- [Running the Management Interface](#)
- [Running the Search Engine](#)

### **Running the WebService**

#### **Running – WebService**

Below are instructions for starting and stopping the WebService on both Windows and Linux.

#### **Windows**

To start the CleanSpeak WebService on Windows when it is not installed as a Windows Service, execute the startup.bat script located in the CLEANSPEAK\_HOME/apache-tomcat-7.0.16/bin directory using the command-line like this:

```
> startup.bat
```

To stop the CleanSpeak WebService on Windows, execute the shutdown.bat script like this:

```
> shutdown.bat
```

You can also start and stop CleanSpeak by double clicking the these scripts from Windows Explorer.

#### **Windows Service**

If you installed the CleanSpeak WebService as a Windows Service, you can configure it to automatically be started when Windows boots up. This

configuration is part of the Windows Services Administration tool. Consult Windows help to learn how to access this tool. You can also manually start and stop the CleanSpeak products from this tool as well.

### **Linux (RPM and Deb bundles)**

These packages both include an init script that is used to control the application as well as start the application when the server is booted up. To start the CleanSpeak WebService execute this command:

```
$ sudo /etc/init.d/cleanspeak-webservice start
```

To stop the CleanSpeak WebService execute this command:

```
$ sudo /etc/init.d/cleanspeak-webservice stop
```

### **Linux**

To start the CleanSpeak WebService on Linux when it is not installed using the RPM or Debian packages, execute the startup.sh script located in the CLEANSPEAK\_HOME/apache-tomcat-7.0.16/bin directory like this:

```
$ ./startup.sh
```

To stop the CleanSpeak WebService on Linux, execute the shutdown.sh script like this:

```
$ ./shutdown.sh
```

Once you have the WebService up and running, you can do a number of different things with it:

- Verify it is working with the **/test** page
- Look at the performance metrics with the **/metrics** page
- Send requests to the WebService

### **Testing**

To test that the WebService is running correctly, open this URL in your browser:

<http://localhost:8001/test>

This page contains a form that allows you to test the WebService to ensure it is working properly.

### **Metrics**

Each WebService instance provides a page that displays the current performance metrics and some historical data. You can view this information by opening this URL in your browser:

<http://localhost:8001/metrics>

## **Running the Management Interface**

### **Running – Management Interface**

Below are instructions for starting and stopping the Management Interface on both Windows and Linux.

### **Windows**

To start the CleanSpeak Management Interface on Windows when it is not installed as a Windows Service, execute the startup.bat script located in the CLEANSPEAK\_HOME/apache-tomcat-7.0.16/bin directory using the command-line like this:

```
> startup.bat
```

To stop the CleanSpeak Management Interface on Windows, execute the shutdown.bat script like this:

```
> shutdown.bat
```

You can also start and stop the CleanSpeak Management Interface by double clicking the these scripts from Windows Explorer.

### **Windows Service**

If you installed the CleanSpeak Management Interface as a Windows Service, you can configure it to automatically be started when Windows boots up. This configuration is part of the Windows Services Administration tool. Consult Windows help to learn how to access this tool. You can also manually start and stop the CleanSpeak products from this tool as well.

### **Linux (RPM and Deb bundles)**

These packages both include an init script that is used to control the application as well as start the application when the server is booted up. To start the CleanSpeak Management Interface execute this command:

```
$ sudo /etc/init.d/cleanspeak-management-interface start
```

To stop the CleanSpeak™ Management Interface execute this command:

```
$ sudo /etc/init.d/cleanspeak-management-interface stop
```

### **Linux**

To start the CleanSpeak Management Interface on Linux when it is not installed using the RPM or Debian packages, execute the startup.sh script located in the CLEANSPEAK\_HOME/apache-tomcat-7.0.16/bin directory like this:

```
$ ./startup.sh
```

To stop the CleanSpeak Management Interface on Linux, execute the shutdown.sh script like this:

```
$ ./shutdown.sh
```

### **Testing**

Once you have started the Management Interface, you can test that it is working properly by opening this URL in your browser:

<http://localhost:8011/>

You might need to modify the port if you changed the port Tomcat listens on in the Tomcat configuration file.

## **Running the Search Engine**

### **Running – Search Engine**

Below are instructions for starting and stopping the Search Engine on both Windows and Linux.

### **Windows**

To start the CleanSpeak Search Engine on Windows when it is not installed as a Windows Service, execute the startup.bat script located in the CLEANSPEAK\_HOME/apache-tomcat-7.0.16/bin directory using the command-line like this:

```
> startup.bat
```

To stop the CleanSpeak Search Engine on Windows, execute the shutdown.bat script like this:

```
> shutdown.bat
```

You can also start and stop the CleanSpeak Search Engine by double clicking the these scripts from Windows Explorer.

### **Windows Service**

If you installed the CleanSpeak Search Engine as a Windows Service, you can configure it to automatically be started when Windows boots up. This configuration is part of the Windows Services Administration tool. Consult Windows help to learn how to access this tool. You can also manually start and stop the CleanSpeak products from this tool as well.

### **Linux (RPM and Deb bundles)**

These packages both include an init script that is used to control the application as well as start the application when the server is booted up. To start the CleanSpeak Search Engine execute this command:

```
$ sudo /etc/init.d/cleanspeak-search-engine start
```

To stop the CleanSpeak Search Engine execute this command:

```
$ sudo /etc/init.d/cleanspeak-search-engine stop
```

### **Linux**

To start the CleanSpeak Search Engine on Linux when it is not installed using the RPM or Debian packages, execute the startup.sh script located in the CLEANSPEAK\_HOME/apache-tomcat-7.0.16/bin directory like this:

```
$ ./startup.sh
```

To stop the CleanSpeak Search Engine on Linux, execute the shutdown.sh script like this:

```
$ ./shutdown.sh
```

## **Scaling and Redundancy**

### **CleanSpeak Scaling and Redundancy**

This chapter provides instructions when it becomes necessary to scale CleanSpeak to multiple servers or to provide redundancy for CleanSpeak. In general, scaling CleanSpeak requires the deployment of multiple WebService and Search Engine servers. Unless the Management Interface is a vital part of your process, you will only need one server running it. However, if you use the Management Interface heavily and prefer to have redundancy for that application, you can run multiple instances of it as well.

Here are instructions for deploying multiple servers and configuring them to provide scaling and redundancy for each of the applications that are part of CleanSpeak.

- [Scaling the WebService](#)
- [Scaling the Management Interface](#)

- [Scaling the Search Engine](#)

## Scaling the WebService

### Scaling – WebService

Below are instructions to deploy multiple instances of the CleanSpeak WebService for performance, scaling and redundancy.

#### **Stateless**

The WebService is completely stateless. This means that you can deploy as many instances of it as you need. If you deploy multiple instances of the WebService and implement an effective load-balancer, you will have both a high performance system as well as redundancy in case a single server fails.

#### **Load Balancing**

After all of the WebService instances are deployed, you will need to setup a load-balancer. There are many options for load balancing including a hardware load-balancer such as Barracuda or a software load-balancer such as Apache.

Since the WebService is stateless, the load-balancer does not need to perform session pinning or replication at all. It can also use a simple round-robin or random selection process to choose the next WebService server to forward requests to.

Selection, installation, configuration, and maintenance of a load-balancer is out of scope of this documentation. If you need assistance with selecting, installing, configuring or maintaining a load-balancer, [contact Inversoft Support](#) for more information about our services.

You can also find a wealth of information on load-balancing online.

## Scaling the Management Interface

### Scaling – Management Interface

Below are instructions to deploy multiple instances of the CleanSpeak Management Interface for performance, scaling and redundancy.

#### **Stateful**

The Management Interface maintains a session for each user once you log in to the application. Therefore, if you decide to use multiple instances of the Management Interface for redundancy and scaling, you will need to implement a load-balancer that is stateful.

#### **Load Balancing**

After all of the Management Interface instances are deployed, you will need to setup a load-balancer. There are many options for load balancing including a hardware load-balancer such as Barracuda or a software load-balancer such as Apache.

Since the Management Interface is stateful, the load-balancer will need to perform session pinning (also called session persistence) using a cookie or some other mechanism. Most load-balancers are capable of providing pinning for HTTP applications. You should consult the documentation for your load-balancer to determine how to enable and configure session pinning.

Selection, installation, configuration, and maintenance of a load-balancer is out of scope of this documentation. If you need assistance with selecting, installing, configuring or maintaining a load-balancer, [contact Inversoft Support](#) for more information about our services.

You can also find a wealth of information on load-balancing online.

## Scaling the Search Engine

### Scaling – Search Engine

Below are instructions to deploy multiple instances of the CleanSpeak Search Engine for performance, scaling and redundancy.

#### **Stateless**

The Search Engine is completely stateless. This means that you can deploy as many instances of it as you need. If you deploy multiple instances of the Search Engine and implement an effective load-balancer, you will have both a high performance system as well as redundancy in case a single server fails.

Each instance of the Search Engine will maintain its own database. This is known as sharding or distributed storage. CleanSpeak is designed to handle this situation and still provide accurate search results. It is therefore extremely important that you specify each Search Engine in the Settings section of the Management Interface.

## Load Balancing

You do not need to load-balance the Search Engine servers at all. CleanSpeak handles load-balancing between all of the servers automatically for you as long as you correctly specify all the servers via the Management Interface.

## WebServices

### CleanSpeak WebServices

This chapter covers the following WebServices APIs that are provided by CleanSpeak.

- [Content Item: Filtering Only](#)
- [Content Item: Moderation](#)
- [Content User](#)
- [System User](#)

All of these APIs are accessed through the CleanSpeak Webservice application. This section assumes that you have that application correctly installed and running.

### WebServices – Overview

The CleanSpeak Webservice is built on RESTful, HTTP architecture where the client (your application) makes HTTP requests on the Webservice server to perform specific functions within the CleanSpeak system. Each HTTP request represents the state of the particular function to be performed and is represented as follows:

- **URI:** Where are you performing the function? (e.g. /content/user)
- **HTTP Method:** What operation (or type of function) are you performing? (e.g. HTTP GET).
- **Request Parameters:** How are you performing the operation (e.g. ?uid=foo)

#### HTTP Method

Each operation for a REST Webservice URI is based on the HTTP method that is used. The CleanSpeak Webservice provides for the four, standard, CRUD-based operations: Create (POST), Read (GET), Update (PUT), Delete (DELETE).

#### Request Parameters

Request parameters are sent to the REST Webservices using the HTTP standard that most browsers conform to. Each HTTP method receives parameters differently in order to conform to this standard. Here are each of the HTTP methods and how they receive parameters:

Method	Parameters
GET	<p>Parameters for a GET request are passed as URL parameters within the HTTP header. These parameters are define as part of the URL specification that HTTP uses. That specification is located at <a href="http://www.ietf.org/rfc/rfc2396.txt">http://www.ietf.org/rfc/rfc2396.txt</a>. Depending on the Webservice, most GET requests can take an ID parameter. This ID parameter can be passed as a URL parameter or part of the URI. Look at the documentation page for the specific Webservice to determine if it contains an ID parameter. Here are some examples of GET URLs:</p> <ul style="list-style-type: none"><li>• <a href="http://localhost:8001/example/1">http://localhost:8001/example/1</a></li><li>• <a href="http://localhost:8001/example?id=1">http://localhost:8001/example?id=1</a></li><li>• <a href="http://localhost:8001/example?search=some+search+string">http://localhost:8001/example?search=some+search+string</a></li></ul>
POST	<p>Parameters for a POST request can be passed in the HTTP message-body (also known as the entity-body) or as URL parameters. The ID parameter might also be passed as part of the URI, depending on the specific Webservice you are calling. The HTTP message-body is defined in the HTTP specification here: <a href="http://www.w3.org/Protocols/rfc2616/rfc2616-sec4.html#sec4.3">http://www.w3.org/Protocols/rfc2616/rfc2616-sec4.html#sec4.3</a>. The format of the message-body is specified by the HTTP Transfer-Encoding, which is specified by the <b>Content-Type</b> HTTP header. To use the message-body mechanism to send in the data you must set the <b>Content-Type</b> header to <b>application/x-www-form-urlencoded</b>. If you want more information on the <b>application/x-www-form-urlencoded</b> Transfer-Encoding it is defined as part of the HTML specification here: <a href="http://www.w3.org/TR/html401/interact/forms.html">http://www.w3.org/TR/html401/interact/forms.html</a>. Here is an example of HTTP bodies for POST requests:</p> <ul style="list-style-type: none"><li>• <code>contentItem.content=testing&amp;filter.enabled=true&amp;filter.blacklist.enabled=true</code></li></ul>
PUT	<p>Parameters for a PUT request are the same as the POST request for consistency, even though the HTML specification does not cover the PUT method.</p>

DELETE	Parameters for a DELETE request are the same as the GET request. Similarly, the ID parameter can be set as a URL parameter or part of the URI.
--------	--

## Content Item: Filtering Only

- [Content Item Webservice](#)
  - [Communicating With CleanSpeak](#)
    - [Coding the Client](#)
    - [Request URLs](#)
    - [HTTP Method](#)
    - [Response codes](#)
  - [Filtering Content](#)
    - [Request Parameters](#)
      - [CleanSpeak Database Filter Parameters](#)
      - [Character Filter Parameters](#)
      - [Email Filter Parameters](#)
      - [Phone Number Filter Parameters](#)
      - [URL Filter Parameters](#)
      - [Word Filter Parameters](#)
  - [Response](#)
    - [JSON Response](#)
    - [XML](#)
  - [Example Code](#)

## Content Item Webservice

The Content Item service provides a single API that performs all content handling functionality. This includes filtering (for emails, URLs, and all entries on the white and black lists in the database), content storage and search indexing, content moderation, and content updating. This section specifically covers filtering only. If you've licensed both the filter and moderation components please also read the [Content Item: Moderation](#) section after reading this section.

Based on the parameters you pass to the Webservice and the configuration in the database, CleanSpeak might perform one or more operations during each request. Refer to the [Filtering Concepts](#) guide or coordinate with the project manager to determine the operations that will be performed.

### Communicating With CleanSpeak

#### *Coding the Client*

To filter content, you must communicate with CleanSpeak by implementing code in your client that performs a RESTful HTTP request on the CleanSpeak Webservice. A general overview is in the [WebServices](#) overview section. Filtering details are provided below.

#### *Request URLs*

The URL for the content handling API determines the response type. Here are the URLs and their associated response types:

URL	Response Type
<a href="http://localhost:8001/content/item.js">http://localhost:8001/content/item.js</a>	JSON response
<a href="http://localhost:8001/content/item.xml">http://localhost:8001/content/item.xml</a>	XML response
<a href="http://localhost:8001/content/item">http://localhost:8001/content/item</a>	XML response

#### *HTTP Method*

In order to perform a filtering request, you must use the POST method.

#### *Response codes*

You can determine if the request was successful or not based on the response code. Here are the response codes and their meanings:

Code	Description
200	The request was successful.



400	The request was invalid and malformed. The response will contain a JSON or XML message with the specific errors using the standard CleanSpeak™ error response. That response is covered in the <a href="#">Standard Error Response</a> document.
401	WebService authentication is enabled and you did not supply a valid Authentication header. The response will be empty.

## Filtering Content

The content item API will execute real-time filtering depending on the parameters passed in the request. To turn on filtering, you must specify at least these parameters in your request:

Parameter	Value	Filter
contentItem.content	A UTF-8 encoded string	The content to filter
contentItem.type	text	set to 'text' to configure CleanSpeak to filter incoming content that's text
filter.operation	replace, match, locate	The filtering operation to perform

Additionally, you must enable at least one filter. The method for enabling filters is discussed below.

## Request Parameters

As we mentioned above, you must specify the **contentItem.content**, **contentItem.type**, **filter.operation** and turn on one filter. Before we discuss turning on filters, let's cover the filter operation types that you can specify for the **filter.operation** parameter. There are currently three operations that CleanSpeak can take when it finds something in your content. These are:

Name	Description
match	The match operation tells the filter to find and return whether or not there were any matches. This result is a simple true or false <a href="#">response</a> from the WebService. The match operation doesn't take any additional attributes, but you may optionally tell the filter the types of content to locate using additional parameters that are described below.
locate	The locate operation tells the filter to find and return the location of any matches it finds in the content. The <a href="#">response</a> from the WebService will contain these results. The locate operation doesn't take any additional attributes, but you must tell the filter the types of content to locate using additional parameters that are described below.
replace	The replace operation tells the filter to find and replace all matches inside the content with either a character (such as *) or another String (such as "censored"). The <a href="#">replace result</a> is returned from the WebService as a String. The replace operation has two optional parameters that control how the matches are replaced. These parameters are named replaceChar and replaceString (described below). The default replacement is the "*" character.

Next, you need to tell the Real-Time Filter what to look for in the content. To do this, you must enable each filter you want to use. There are a number of different filter types that CleanSpeak provides. These are:

- **Blacklist** – This tells the Real-Time Filter to filter all of the words and phrases against the blacklist in the database. You can specify severity levels, tags, and other attributes with additional parameters [described below](#). You can manage your white and black lists using the [Management Interface](#)
- **Characters** – This tells the Real-Time Filter to filter out specific characters. The characters to filter are specified using an additional parameter
- **Email** – This tells the Real-Time Filter to filter email addresses. The filter will attempt to find anything that looks like a valid email address inside the content
- **Phone Numbers** – This tells the Real-Time Filter to filter phone numbers. The filter will attempt to find anything that looks like a valid phone number inside the content. You can also use this filter to handle other numeric values including social security numbers
- **URLs** – This tells the Real-Time Filter to filter URLs. The filter will attempt to find anything that looks like a valid URL inside the content
- **Words** – This tells the Real-Time Filter to filter out specific words. The words to filter are specified using an additional parameter

To turn on each filter, you must specify different parameters. Here are the parameters and the filter they control:

Parameter	Value	Filter
filter.blacklist.enabled	true	The CleanSpeak Database filter
filter.characters.enabled	true	The Character filter
filter.emails.enabled	true	The Email filter
filter.phoneNumbers.enabled	true	The Phone Number filter

filter.urls.enabled	true	The URL filter
filter.words.enabled	true	The Word filter

Here is a simple example using Java to access the Webservice and request a Real-Time Filter replace operation. This request will enable the blacklist filter and tell it to perform a replace operation:

```

URL url = new URL("http://localhost:8001/content/item.js");
URLConnection urlConnection = (URLConnection) url.openConnection();
urlConnection.setDoInput(true);
urlConnection.setDoOutput(true);
urlConnection.setRequestMethod("POST");
urlConnection.addRequestProperty("Content-Type",
"application/x-www-form-urlencoded");
urlConnection.connect();

OutputStream out = urlConnection.getOutputStream();
OutputStreamWriter writer = new OutputStreamWriter(out);
writer.append("contentItem.content=").append(URLEncoder.encode("Some
content", "UTF-8")).append("&");
writer.append("contentItem.type=text&");
writer.append("filter.operation=replace&");
writer.append("filter.blacklist.enabled=true&");
writer.flush();

InputStream in = urlConnection.getInputStream();
// Process the input stream for the XML or JSON here

```

The specific parameters for each filter type are covered below.

Lastly, you can control the **replace** operation using two additional parameters. Those parameters are:

Name	Description	Type
filter.replaceChar	The character that is used during a replace operation	char
filter.replaceString	The String that is used during a replace operation	String

#### CleanSpeak Database Filter Parameters

This filter searches for the entries on your blacklist. You can control which words the filter looks for using a number of different parameters. To determine which parameters to set, contact the project manager or review the [Filtering Concepts](#) and [List Management](#) pages. Here are the parameters that this filter accepts:

Name	Description	Type	Required?
filter.blacklist.enabled	Turns on the CleanSpeak Database Filter.	boolean	Yes
filter.blacklist.tag	This parameter controls which entries on your blacklist are filtered against based on the tags you have assigned to the entries. You can specify multiple tags by including this parameter multiple times. By default, CleanSpeak filters against all tags. Default tags are described in the <a href="#">List Management</a> section.	String	No

filter.blacklist.locale	<p>This parameter controls which languages the filter will look for. Each word on the blacklist in your database has a locale defined. This parameter will instruct the filter to only look for entries on the blacklist with the certain locales. You can specify multiple locales by supplying this parameter multiple times. If left undefined, CleanSpeak will filter against all locales you have stored in your database.</p> <p>The format for locale is two digit lower case <a href="#">ISO 639-1 Language Code</a> followed by an optional underscore and <b>upper case</b> two digit <a href="#">ISO 3166 country code</a>.</p> <p>Examples:</p> <pre> filter.blacklist.locale=en // All English entries filter.blacklist.locale=en_US // Only American-English entries filter.blacklist.locale=es // All Spanish entries filter.blacklist.locale=es_MX // Only Mexican-Spanish entries </pre>	String	No
filter.blacklist.severity	<p>This parameter controls which entries on your blacklist are filtered against based on the severity assigned to them. The filter will search for words and phrases on the Blacklist whose severity setting is equal to or higher than than this value. For example, if you specify <b>high</b> the filter will search for words and phrases marked as <b>high</b> or <b>severe</b>.</p>	String	No

#### Character Filter Parameters

This filter searches for a set of characters. You can control the characters the filter searches for using a request parameter. Here are the parameters that this filter accepts:

Name	Description	Type	Required?
filter.characters.enabled	Turns on the Characters filter.	boolean	Yes
filter.characters.character	<p>This parameter specifies the characters that should be searched for. You should specify a single character for this parameter. You can specify this parameter multiple times to filter multiple characters. Here is an example of using this parameter multiple times:</p> <pre> filter.characters.character=@&amp; filter.characters.character=* </pre>	char	Yes

#### Email Filter Parameters

This filter searches for email addresses. When the filter finds a match, it will associate a quality score with the match between 0 and 1, 1 being the highest. Examples: "foo at this . it" returns a quality score of 0.40; "foo@inversoft.com" returns a quality score of 1.0. The logic to determine the quality score can be adjusted with the optional parameters below. **NOTE:** It is recommended to use the **locate** filter operation with a response handler in order to set the quality score threshold that is appropriate for your application. When using **match** or **replace**, the filter will return a positive match on any quality score greater than 0.

Name	Description	Type	Required?
filter.emails.enabled	Turns on the Email filter.	boolean	Yes
filter.emails.maximumMatchLength	This parameter controls the maximum length that a match can be in order to be considered an email. This defaults to 50	int	No
filter.emails.spacePenalty	This parameter specifies a penalty applied if the match contains any spaces. For example, <b>foo @bar.com</b> contains three spaces. This defaults to -0.05	double	No

filter.emails.domainQuality.domain	<p>This parameter specifies the initial quality score for a specific domain. For example, you can set the initial quality for the domain <b>it</b> to 0.5 by setting the <b>domain</b> parameter to <b>it</b> and the <b>quality</b> parameter to 0.5. The defaults for domain qualities are too numerous to list here, but most dictionary words default to 0.5 and everything else defaults to 1.0. You can set multiple domain qualities by specifying this parameter multiple times using an array index like this:</p> <pre>filter.emails.domainQuality[0].domain=it filter.emails.domainQuality[0].quality=0.5 filter.emails.domainQuality[1].domain=com filter.emails.domainQuality[1].quality=0.9</pre>	String	No
filter.emails.domainQuality.quality	<p>This parameter specifies the initial quality score for a specific domain. For example, you can set the initial quality for the domain <b>it</b> to 0.5 by setting the <b>domain</b> parameter to <b>it</b> and the <b>quality</b> parameter to 0.5. The defaults for domain qualities are too numerous to list here, but most dictionary words default to 0.5 and everything else defaults to 1.0. You can set multiple domain qualities by specifying this parameter multiple times using an array index like this:</p> <pre>filter.emails.domainQuality[0].domain=it filter.emails.domainQuality[0].quality=0.5 filter.emails.domainQuality[1].domain=com filter.emails.domainQuality[1].quality=0.9</pre>	double	No

### Phone Number Filter Parameters

This filter searches for phone numbers (or other types of numbers). When the filter finds a match, it will associate a quality score with the match between 0 and 1, 1 being the highest. Examples: "303; ;555; ;onetwothreefour" returns a quality score of 0.63; "3035551234" returns a quality score of 1.0. The logic to determine the quality score can be adjusted with the optional parameters below. **NOTE:** It is recommended to use the **locate** filter operation with a response handler in order to set the quality score threshold that is appropriate for your application. When using **match** or **replace**, the filter will return a positive match on any quality score greater than 0.

Name	Description	Type	Required?
filter.phoneNumbers.enabled	Turns on the Phone Number filter.	boolean	Yes
filter.phoneNumbers.maximumMatchLength	This parameter specifies the maximum length that a match can be in order to be considered a phone number. This defaults to 20 in order to cover most world wide phone number formats.	int	No
filter.phoneNumbers.minimumMatchLength	This parameter specifies the minimum length that a match can be in order to be considered a phone number. This defaults to 6 in order to cover most world wide phone number formats.	int	No
filter.phoneNumbers.separatorPenalty	This parameter specifies a penalty that is applied to the quality score for a match if it contains any type of separator other than a dash or parenthesis. For example, <b>303;555;1234</b> contains two penalized separators. This defaults to -0.05	double	No
filter.phoneNumbers.spacePenalty	This parameter specifies a penalty that is applied to the quality score for a match if it contains a space. For example, <b>303 555 1234</b> contains two spaces. This defaults to -0.05	double	No
filter.phoneNumbers.wordPenalty	This parameter specifies a penalty that is applied to the quality score for a match if it contains any words rather than digits. For example, <b>three zero three 555 1234</b> contains three words. This defaults to -0.03	double	No

### URL Filter Parameters

This filter searches for URLs. When the filter finds a match, it will associate a quality score with the match between 0 and 1, 1 being the highest. Examples: "this. it" returns a quality score of 0.45; "inversoft.com" returns a quality score of 1.0. The logic to determine the quality score can be adjusted with the optional parameters below. **NOTE:** It is recommended to use the **locate** filter operation with a response handler in order to set the quality score threshold that is appropriate for your application. When using **match** or **replace**, the filter will return a positive match on any quality score greater than 0.

Name	Description	Type	Required?
filter.urls.enabled	Turns on the URL filter.	boolean	Yes
filter.urls.maximumMatchLength	This parameter controls the maximum length that a match can be in order to be considered a URL. This defaults to 50	int	No
filter.urls.spacePenalty	This parameter specifies a penalty applied if the match contains any spaces. For example, <b>www . example . com</b> contains four spaces. This defaults to -0.05	double	No

filter.urls.domainQuality.domain	<p>This parameter specifies the initial quality score for a specific domain. For example, you can set the initial quality for the domain <b>it</b> to 0.5 by setting the <b>domain</b> parameter to <b>it</b> and the <b>quality</b> parameter to 0.5. The defaults for domain qualities are too numerous to list here, but most dictionary words default to 0.5 and everything else defaults to 1.0. You can set multiple domain qualities by specifying this parameter multiple times using an array index like this:</p> <pre>filter.urls.domainQuality[0].domain=it filter.urls.domainQuality[0].quality=0.5 filter.urls.domainQuality[1].domain=com filter.urls.domainQuality[1].quality=0.9</pre>	String	No
filter.urls.domainQuality.quality	<p>This parameter specifies the initial quality score for a specific domain. For example, you can set the initial quality for the domain <b>it</b> to 0.5 by setting the <b>domain</b> parameter to <b>it</b> and the <b>quality</b> parameter to 0.5. The defaults for domain qualities are too numerous to list here, but most dictionary words default to 0.5 and everything else defaults to 1.0. You can set multiple domain qualities by specifying this parameter multiple times using an array index like this:</p> <pre>filter.urls.domainQuality[0].domain=it filter.urls.domainQuality[0].quality=0.5 filter.urls.domainQuality[1].domain=com filter.urls.domainQuality[1].quality=0.9</pre>	double	No

### Word Filter Parameters

This filter searches for an arbitrary set of words. The words it searches for are specified using a request parameter. Here are the parameters that this filter accepts:

Name	Description	Type	Required?
filter.words.enabled	Turns on the Word filter.	boolean	Yes
filter.words.word	<p>This parameter specifies a word that the filter will search for. You can specify multiple words by supplying this parameter multiple times. For example:</p> <pre>filter.words.word=foo&amp; filter.words.word=bar</pre>	String	Yes if the word filter is enabled

### Response

There are two types of responses that are sent back from the Real-Time Filter API, XML and JSON. In general, you will receive better performance using JSON due to the lower parsing overhead. Here are the formats of the XML and JSON responses.

#### JSON Response

**JSON Response**

```
{
  "filter": {
    "matched": true,
    "matches": [
      { "start": 0, "length": 6, "matched": "fucker", "type":
"cleanspeakdb",
        "root": "fuck", "tags": ["Slang"], "severity": "severe" },
      { "start": 7, "length": 4, "matched": "shit", "type": "cleanspeakdb",
        "root": "shit", "tags": ["Slang"], "severity": "severe" }
    ],
    "replacement": "***** ****"
  }
}
```

The JSON keys and values are described in this table:

Key	Value Description
filter	The filter object that contains the result from a filter request.
filter.matched	A boolean that defines if the filter found any results or not.
filter.matches	A list of matches that the filter found if the operation is set to <b>locate</b> .
filter.matches.tags	If the match is of the type <b>blacklist</b> then this will contain an array that contains all of the tags of the blacklist entry that was matched.
filter.matches.length	The length of a single match.
filter.matches.matched	If the match is of the type <b>blacklist</b> then this will contain the conjugation or variation that the filter matched on.
filter.matches.root	If the match is of the type <b>blacklist</b> then this will contain the root word of the match.
filter.matches.severity	If the match is of the type <b>blacklist</b> then this will contain the severity of the match.
filter.matches.start	The start index of a single match.
filter.matches.type	The type of the match. This will correspond to the filter types that were enabled and configured in the request. The current filters that the CleanSpeak™ Real-Time Filter supports are: <ul style="list-style-type: none"><li>• characters</li><li>• cleanspeakdb</li><li>• emails</li><li>• phoneNumbers</li><li>• urls</li><li>• words</li></ul>
filter.replacement	The content with all matches replaced if the operation is set to <b>replace</b> .

## XML

The XML response is nearly identical to the JSON response. The main differences are that the element names are not plural (match instead of matches) and the tags of each **match** element are returned as sub-elements. Here is an example XML response:

```
<?xml version="1.0" encoding="UTF-8"?>
<response
xmlns="http://www.inversoft.com/schemas/cleanspeak/content/item/2">
  <filter matched="true">
    <match start="0" length="6" type="blacklist" locale="en"
matched="fucker" root="fuck" severity="severe">
      <tag>Slang</tag>
    </match>
    <replacement><![CDATA[*****]]>
  </filter>
</response>
```

## Example Code

Here is a simple example using Java to access the WebService and request a Real-Time Filter replace operation. This will turn on the Real-Time Filter, tell it to perform a replace operation, and apply the cleanspeakdb and url filters:

```

URL url = new URL("http://localhost:8001/content/item.js");
URLConnection urlConnection = (URLConnection) url.openConnection();
urlConnection.setDoInput(true);
urlConnection.setDoOutput(true);
urlConnection.setRequestMethod("POST");
urlConnection.addRequestProperty("Content-Type",
"application/x-www-form-urlencoded");
urlConnection.connect();

OutputStream out = urlConnection.getOutputStream();
OutputStreamWriter writer = new OutputStreamWriter(out);
writer.append("contentItem.content=").append(URLEncoder.encode("Some
content", "UTF-8")).append("&");
writer.append("contentItem.type=text&");
writer.append("filter.operation=replace&");
writer.append("filter.blacklist.enabled=true&");
writer.append("filter.urls.enabled=true");
writer.flush();

InputStream in = urlConnection.getInputStream();
// Process the input stream for the XML or JSON here

```

For more examples, refer to the [Examples](#) page.

## Content Item: Moderation

### Moderation

#### Communicating with CleanSpeak

- [Coding the Client](#)
- [Moderation Request](#)
  - [More on 'contentItem.receiverUID'](#)
- [Moderation Response](#)
  - [JSON](#)
  - [XML](#)
- [Example Code](#)

#### Updating Content

- [Content Item UID](#)

#### Image Moderation

## Moderation

If you have licensed the CleanSpeak Moderation tools, you can utilize the Content Item service to store and monitor your user generated content via the CleanSpeak Management Interface. Coordinate with the project manager or refer to the [Moderation Concepts](#) and [Moderation Setup & Configuration](#) pages to configure the Management Interface prior to implementing the request parameters below.

## Communicating with CleanSpeak

### Coding the Client

Once you've [configured your applications and components](#) via the Management Interface, it's now time to implement the code in your client that will communicate with CleanSpeak. Like filtering, this communication process occurs via a RESTful HTTP request that you make on the CleanSpeak Webservice. In most cases, the request that you use to filter content is the SAME request for moderating content. This means that you will be sending ONE single request to CleanSpeak to handle both filtering and moderating of content. The only difference is that when you're moderating, you must provide the additional parameters described below in addition to the parameters describe in the [Content Item: Filtering Only](#) section.

## Moderation Request

The following table provides information on each parameter used when performing content moderation. For specifics for how to build your RESTful HTTP request, please reference the [Content Item: Filtering](#) section described earlier.

Name	Description	Type	Required?
contentItem.application	The name of the Application that the content was generated in. This must match the name you setup in the Management Interface exactly.	String	Yes
contentItem.component	The name of the Component that the content was generated in. This must match the name you setup in the Management Interface exactly.	String	Yes
contentItem.type	In order for the content to be searchable, this parameter must be set to <b>text</b> . All other types of content can be stored and potentially moderated, but they will not be searchable.	String	Yes
contentItem.createInstant	This parameter indicates the exact timestamp when the content was generated within the Application and Component. This must be an integer value that represents the number of milliseconds that the content was generated since Unix standard Epoch of January 1, 1970 UTC.  An example value is <b>1314066305361</b> (this is equivalent to roughly Tuesday August 23, 2011 02:27:18 UTC)	Integer	Yes
contentItem.uid	This is an optional parameter that your system generates that uniquely identifies a piece of content in your system. This property is required if you need to make updates to an existing piece of content, are using the approval queue moderation system, or are making updates (edit or delete) on content from within the Management Interface.  More information on the approval queue system can be found in our <a href="#">Content Moderation</a> documentation. Additionally, if you would like more information on updating content, refer to our <a href="#">Content Updating</a> documentation below.  <b>NOTE:</b> This UID must be globally unique across all of your Applications and Components.	String	Yes, if using the approval queue.
contentItem.location	This is an Application and Component specific location identifier. For example, you might use an area ID for a game or a thread ID for a forum. This parameter is used by CleanSpeak™ to build <a href="#">threaded views</a> of the content.	String	No
contentItem.receiverUID	This is a unique identifier for the person that received the content. This should only be used for Private Messaging between two users. You can provide information about your users to CleanSpeak™ using the Content Users API. That API is covered in the <a href="#">Content Users</a> document.  <b>NOTE:</b> This ID must be unique across all of your Applications and Components.	String	No
contentItem.senderUID	This is unique identifier for the person that generated the content. This parameter is used by CleanSpeak™ to associate the content with the user that generated it. It is not required, but if you know the user's ID, it is highly recommended that you send it to CleanSpeak™. You can provide additional information about your users (such as name, last login, etc) to CleanSpeak™ using the Content Users API. That API is covered in the <a href="#">Content Users</a> document.  <b>NOTE:</b> This ID must be unique across all of your Applications and Components.	String	No
moderation.type	If sending content to an approval queue, this parameter must be set to the value <b>requiresApproval</b> in order for the content to be added to the approval moderation queue. See the <a href="#">Example Deployment</a> section of the moderation user guide for details.	String	Yes, if using the approval queue.

### More on 'contentItem.receiverUID'

As mentioned in the table above, 'contentItem.receiverUID' should only ever be used when there's a private, one-on-one communication between two of your users, which is typically called a 'whisper', 'tell', or PM in most platforms.

Additionally, it should be used in conjunction with a special 'contentItem.location' that's composed by alphabetizing both names and then concatenating them together. Creating a location in this manner is very important when moderating because it allows moderators to view the conversation in context between two users. Review the [Threaded View](#) page for details.

Here's a simple 3-step conversation between userX and userY to help explain:

1. userX chat console:  
/tell userY: Hey, how are you?

2. userY chat console:  
userX: Hey, how are you?  
/tell userX: I'm great, how are you?

3. userX chat console:  
userY: I'm great, how are you?

Needless to say, this is a private conversation occurring between userX and userY. From a code point of view, you'd send information to CleanSpeak as follows:



Step 1:

- userX tells userY "Hey, how are you?"

```
contentItem.content=Hey, how are you?  
contentItem.senderUID=userX  
contentItem.receiverUID=userY  
contentItem.location=userX_userY
```

Step 2:

- userY tells userX "I'm great, how are you?"

```
contentItem.content=I'm great, how are you?  
contentItem.senderUID=userY  
contentItem.receiverUID=userX  
contentItem.location=userX_userY
```

Notice the senderUID and receiverUID alternate depending on which user is generating the content. Moreover, notice that the 'contentItem.location' is built by alphabetizing both usernames and then concatenating them together with an underscore. The 'alphabetizing both usernames and then concatenating them together' is necessary because it uniquely identifies a private conversation between the same two people regardless of who starts the conversation.

## Moderation Response

There are two types of responses that are sent back when storing and moderating content: XML and JSON. In general, you will receive better performance using JSON due to the lower parsing overhead. Here are the formats of the XML and JSON responses.

### JSON

```
{  
  "moderation": {  
    "type": "requiresApproval",  
    "stored": true,  
    "queued": false,  
    "alertsEnabled": "false"  
  }  
}
```

The JSON keys and values are described in this table:

Key	Value Description
moderation	The moderation object that contains the result from a Moderation request.
moderation.type	The type of moderation. Either 'requiresApproval', 'requiresReview', 'generatesAlert', or 'unmoderated'. If 'unmoderated', then the content has not been put the Approval, Alert, or Review queue.
moderation.stored	A boolean that indicates if the content was stored or not.
moderation.queued	A boolean that indicates if the content was added to the approval moderation queue. This concept is covered in the <a href="#">Content Moderation</a> document.
moderation.alertsEnabled	A boolean that indicates if alerts are enabled for the application and component specified in the request.

### XML

The XML response is identical to the JSON response. Also, note that the XML response is in the namespace <http://www.inversoft.com/schema/scleanspeak/content/handle/2>. Here is an example XML response:

```
<?xml version="1.0" encoding="UTF-8"?>
<response
xmlns="http://www.inversoft.com/schemas/cleanspeak/content/item/2">
  <moderation type="requiresApproval" stored="true" queued="false"
alertsEnabled="false"/>
</response>
```

## Example Code

Here is a simple example code using Java to access the Webservice and store content.

```
URL url = new URL("http://localhost:8001/content/item.js");
URLConnection urlConnection = (URLConnection) url.openConnection();
urlConnection.setDoInput(true);
urlConnection.setDoOutput(true);
urlConnection.setRequestMethod("GET");
urlConnection.addRequestProperty("Content-Type",
"application/x-www-form-urlencoded");
urlConnection.connect();

OutputStream out = urlConnection.getOutputStream();
OutputStreamWriter writer = new OutputStreamWriter(out);
writer.append("contentItem.content=").append(URLEncoder.encode("Some
content", "UTF-8")).append("&");
writer.append("contentItem.type=text&");
writer.append("contentItem.application=My+Game&");
writer.append("contentItem.component=chat&");
writer.append("contentItem.createInstant=1314066305361&");
writer.append("contentItem.location=MountainTop&");
writer.append("contentItem.senderUID=fred1432");
writer.append("contentItem.receiverUID=waldo001");
writer.flush();

InputStream in = urlConnection.getInputStream();
// Process the input stream for the XML or JSON here
```

## Updating Content

The Content Item web service provides the capability to update content that you're storing and/or moderating. CleanSpeak provides for this option via the 'contentItem.uid' property.

### Content Item UID

To inform CleanSpeak that content is updatable, associate a unique identifier by defining a 'contentItem.uid' when performing a HTTP POST request. This UID is generated by your system and informs CleanSpeak that the incoming piece of content is a unique piece of content in your system. Once associated, CleanSpeak assumes that this content is now updatable and future requests can be made via HTTP PUT to update an existing piece of content.

Making a request to update content is similar to a normal moderation request with the exception that the 'contentItem.uid' parameter is required and you perform a 'PUT' rather than a 'POST'. When performing an update, the 'contentItem.uid' parameter informs CleanSpeak that the piece of content is updatable and can be uniquely identified by both systems using the Content UID specified.

Name	Description	Type	Required?
contentItem.uid	This parameter informs CleanSpeak that a piece of content is uniquely identified in your system. This effectively informs CleanSpeak™ that a piece of content is updatable.  <b>NOTE:</b> This UID must be globally unique across all of your Applications and Components. We recommend using a UUID for this property but it's not required.	String	Yes

## Image Moderation

The Content Item web service provides the capability to moderate images. In addition to the request parameters discussed above, you must modify the following parameters below to enable image filtering:

Name	Description	Type	Required?
contentItem.type	Set to 'image' to configure CleanSpeak to moderate images. This is different from text based filtering, where you'd set this parameter to 'text'	String	Yes if moderating images
contentItem.content	The URL of your image	URL	Yes if moderation images

CleanSpeak expects the URL provided in the 'contentItem.content' parameter to be a valid resource. What this means is that CleanSpeak does not currently support any functionality to manage uploading and storing image files. As a result, you must implement this functionality on your end prior to using CleanSpeak for image moderation.

## Content User

- [WebServices – Content User](#)
  - [URLs](#)
  - [Response codes](#)
  - [Authentication](#)
  - [Operations](#)
  - [UID Request Parameter](#)
  - [User Request Parameters](#)
  - [More on 'contentUser.uid'...](#)
  - [Modifying Users](#)
  - [Render JSON Response](#)
  - [Render XML Response](#)
  - [Example Code](#)
    - [GET Example](#)
    - [POST Example](#)

### WebServices – Content User

You can use the WebService to provide information about the users that are generating content to be stored in CleanSpeak. In most cases, this WebService is called whenever a user signs up, modifies their profile or deletes their profile. The available operations are:

- Retrieve a user's information
- Create a new user
- Modify an existing user
- Delete a user

### URLs

The URL for this WebService determines the response type. Here are the URLs and their associated response types:

URL	Response Type
<a href="http://localhost:8001/content/user.js">http://localhost:8001/content/user.js</a>	JSON response
<a href="http://localhost:8001/content/user.xml">http://localhost:8001/content/user.xml</a>	XML response
<a href="http://localhost:8001/content/user">http://localhost:8001/content/user</a>	XML response

### Response codes

You can determine if the request was successful or not based on the response code. Here are the response codes and their meanings:

Code	Description
200	The request was successful.
400	The request was invalid and malformed. The response will contain a JSON or XML message with the specific errors
401	WebService authentication is enabled and you did not supply a valid Authentication header. The response will be empty
404	The ID is invalid. The response will be empty

### Authentication

If you have enabled WebService authentication through the Management Interface, you will need to pass in the HTTP header named **Authentication** to the WebService. This header should be set to the same value as the authentication key setup in the Management Interface.

### Operations

Below are the HTTP methods, the operation they perform, the request information, successful response message format, and error response format.

Method	Description	Request	Success Response	Error Response
GET	This retrieves a user's information	UID Request Parameters	<ul style="list-style-type: none"> <li>Render User JSON Response</li> <li>Render User XML Response</li> </ul>	Standard Error Response
POST	This creates a new user	User Request Parameters	No response message.	Standard Error Response
PUT	This updates an existing user	User Request Parameters	No response message.	Standard Error Response
DELETE	This deletes an existing user	UID Request Parameters	No response message.	Standard Error Response

### UID Request Parameter

The GET and DELETE operations use a single parameter to identify the user to be retrieved or deleted. Here is that parameter:

Name	Description	Type	Required?
contentUser.uid	<p>The unique identifier of the user to retrieve or delete. This unique identifier is a value that you pass into the WebService when creating the user.</p> <p><b>NOTE:</b> The Content User UID must be unique across all of your configured Applications and Components.</p>	String	Yes

This parameter must be specified using a URL parameter and cannot be supplied using the URL. Here are examples of valid requests and invalid requests

```

http://localhost:8001/content/user?contentUser.uid=2 - Valid
http://localhost:8001/content/user.js?contentUser.uid=2 - Valid
http://localhost:8001/content/user.xml?contentUser.uid=2 - Valid
http://localhost:8001/content/user/2 - Invalid
http://localhost:8001/content/user/2.js - Invalid
http://localhost:8001/conten/user/2.xml - Invalid

```

### User Request Parameters

The POST and PUT operations use a number of different parameters to control the user information that is used to create or update the user. You can specify these parameters as URL parameters or in the HTTP body using the x-www-form-urlencoded transfer encoding. Here are those parameters:

Name	Description	Type	Required?
contentUser.uid	<p>This specifies the unique identifier for the user. This value must be a unique identifier in your system and is a unique key within the CleanSpeak™ Database (but it is NOT the primary key within the CleanSpeak™ Database).</p> <p><b>NOTE:</b> This must be unique across all of your Applications and Components. Moreover, this parameter MUST be the same value you provide as the 'senderUID' when inputting content via the <a href="#">Content Item: Moderation</a> web service.</p> <p>uid=foo</p>	String	Yes

contentUser.name	This is the full name of the user. Ex:  name=Jane Doe	String	No
contentUser.email	The email address of the user. Ex:  email=jane@doe.com	String	No
contentUser.birthDateInstant	must be a long value that represents the number of milliseconds since Epoch UTC at midnight of the date:  birthDateInstant=62406000000	String	No
contentUser.lastLoginInstant	must be a long value that represents the number of milliseconds since Epoch UTC:  lastLoginInstant=1328046664660	String	No
contentUser.attributes['name']	Free form parameter that specifies a custom attribute for the user. You must specify the name of the attribute inside the square brackets and the value of the attribute as the value of the parameter. You can specify multiple attributes by supplying this parameter multiple times. However, you cannot supply a named attribute multiple times. Here are some examples:  attributes['gender']=F attributes['forumName']=qwerty	String	No

### ***More on 'contentUser.uid'...***

As briefly mentioned above, it's critical that all users are unique across all applications and all components in your system. This unique identity is determined by your system and then provided in the form of a 'uid' to CleanSpeak via the 'contentUser.uid' parameter.

It's also enormously beneficial to provide this same 'uid' when sending content to the [Content Item: Moderation](#) web service in the form of the 'contentItem.senderUID' parameter. By adding users to CleanSpeak via the Content User web service and then associating users to content they generate by supplying the 'contentItem.senderUID' via the Content Item web service, you provide your moderators maximum capability to search on content users generate within your system.

### ***Modifying Users***

When updating users with PUT, The CleanSpeak Content User interface will perform a full update against the submitted and omitted request parameters. This means that if, for instance, you've previously set a birth date but in a future PUT request the birth date is omitted, CleanSpeak assumes you wish to remove the birth date from the user. This holds true for all user parameters including free form attributes. The take away here is to make sure you always send the complete set of user data even if you are only updating a single field.

### ***Render JSON Response***

This JSON response contains the information about the user. This is returned from the GET and POST requests. The reason this is used for POST is to provide the ID of the user back to the caller. This ID is the primary key of the user within the CleanSpeak Database. You don't really need to know this value or store it because the UID is sufficient to locate a Content User.

```
{
  "user": {
    "id": 42,
    "uid": "foo",
    "name": "Jane Doe",
    "email": "jane@doe.com",
    "birthDateInstant": "62406000000",
    "lastLoginInstant": "1328046664660",
    "attributes": {
      "gender": "F",
      "forumName": "qwerty"
    }
  }
}
```

### **Render XML Response**

The XML response is similar to the JSON response except that the attributes are passed back using XML elements and the XML element names are not plural. Also, note that the XML uses the namespace <http://www.inversoft.com/schemas/cleanspeak/content/user/2>. Here is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<response
xmlns="http://www.inversoft.com/schemas/cleanspeak/content/user/2">
  <user id="42" uid="foo" name="Jane Doe" email="jane@doe.com"
birthDate="12/1/1" lastLogin="2011-09-26T20:08:17.640Z">
    <attribute name="gender" value="F"/>
    <attribute name="forumName" value="qwerty"/>
  </user>
</response>
```

### **Example Code**

Here is some sample java code that performs the various requests that are part of the Content User API.

#### **GET Example**

```

URL url = new
URL("http://localhost:8001/content/user.js?contentUser.uid=foo");
URLConnection urlConnection = (URLConnection) url.openConnection();
urlConnection.setDoInput(true);
urlConnection.setDoOutput(true);
urlConnection.setRequestMethod("GET");
urlConnection.connect();

if (urlConnection.getResponseCode() == 200 {
    InputStream in = urlConnection.getInputStream();
    // Process the POST response
} else {
    InputStream in = urlConnection.getErrorStream();
    // Process the error response
}
}

```

#### POST Example

```

URL url = new URL("http://localhost:8001/content/user.js");
URLConnection urlConnection = (URLConnection) url.openConnection();
urlConnection.setDoInput(true);
urlConnection.setDoOutput(true);
urlConnection.setRequestMethod("POST");
urlConnection.addRequestProperty("Content-Type",
"application/x-www-form-urlencoded");
urlConnection.connect();

OutputStream out = urlConnection.getOutputStream();
OutputStreamWriter writer = new OutputStreamWriter(out);
writer.append("contentUser.uid=foo&");
writer.append("contentUser.name=Jane%20Doe&");
writer.append("contentUser.email=jane%40doe.com&");
writer.append("contentUser.birthDateInstant=62406000000&");
writer.append("contentUser.lastLoginInstant=1328046664660&");
writer.append("contentUser.attributes['gender']=M&");
writer.append("contentUser.attributes['forumName']=qwerty");
writer.flush();

if (urlConnection.getResponseCode() == 200 {
    InputStream in = urlConnection.getInputStream();
    // Process the POST response
} else {
    InputStream in = urlConnection.getErrorStream();
    // Process the error response
}
}

```

## System User

You can use the System User Webservice to control the system moderators for the CleanSpeak Management Interface. From here you can manage the users that have access to the CleanSpeak Management Interface and all of their information. The available operations are:

- Retrieve a user's information
- Create a new user
- Modify an existing user
- Delete a user

### URLs

The URL for this Webservice determines the response type. Here are the URLs and their associated response types:

URL	Response Type
<a href="http://localhost:8001/system/user.js">http://localhost:8001/system/user.js</a>	JSON response
<a href="http://localhost:8001/system/user.xml">http://localhost:8001/system/user.xml</a>	XML response
<a href="http://localhost:8001/system/user">http://localhost:8001/system/user</a>	XML response

### Response codes

You can determine if the request was successful or not based on the response code. Here are the response codes and their meanings:

Code	Description
200	The request was successful.
400	The request as invalid and malformed. The response will contain a JSON or XML message with the specific errors
401	Webservice authentication is enabled and you did not supply a valid Authentication header. The response will be empty
404	The ID is invalid. The response will be empty

### Authentication

If you have enabled Webservice authentication through the Management Interface, you will need to pass in the HTTP header named **Authentication** to the Webservice. This header should be set to the same value as the authentication key setup in the Management Interface.

### Operations

Below are the HTTP methods, the operation they perform, the request information, successful response message format, and error response format.

Method	Description	Request	Success Response	Error Response
GET	This retrieves a user's information	ID Request Parameters	<ul style="list-style-type: none"> <li>• Render User JSON Response</li> <li>• Render User XML Response</li> </ul>	<a href="#">Standard Error Response</a>
POST	This creates a new user	User Request Parameters	<ul style="list-style-type: none"> <li>• Render User JSON Response</li> <li>• Render User XML Response</li> </ul>	<a href="#">Standard Error Response</a>
PUT	This updates an existing user	User Request Parameters	No response message.	<a href="#">Standard Error Response</a>
DELETE	This deletes an existing user	ID Request Parameters	No response message.	<a href="#">Standard Error Response</a>

### ID Request Parameters

The GET and DELETE operations use a single parameter to identify the user to be retrieved or deleted. Here is that parameter:

Name	Description	Type	Required?
id	The ID of the user to retrieve or delete	Integer	Yes

You can specify the ID parameter as a URL parameter for the GET and DELETE operations, or you can specify it via the URI. Here are examples of each of those URLs:



```

http://localhost:8001/system/user?id=2

http://localhost:8001/system/user.js?id=2

http://localhost:8001/system/user.xml?id=2

http://localhost:8001/system/user/2

http://localhost:8001/system/user/2.js

http://localhost:8001/system/user/2.xml

```

### User Request Parameters

The POST and PUT operations use a number of different parameters to control the user information that is used to create or update the user. You can specify these parameters as URL parameters or in the HTTP body using the x-www-form-urlencoded transfer encoding. Additionally, you can specify the ID parameter on the URI in the same you can for the GET or DELETE operations. Here are those parameters:

Name	Description	Type	Required?
systemUser.id	This specifies the user to update or the ID to assign to a new user. If you pass this in to a POST request you must ensure that the ID given will not cause unique key violations.	Integer	Yes for GET, PUT and DELETE. No for POST
systemUser.email	The email (or username) for the user	String	Yes
systemUser.password	The password for the user. If the request is a PUT and this is not specified, the user's current password will remain in effect	String	Yes for POST. No for PUT
systemUser.role	The role(s) for the user. You can specify this parameter multiple times, once for each role. Here is an example of specifying multiple roles: <code>role=filter&amp;role=moderator</code>	String	Yes
systemUser.applicationID	The IDs of the Applications the user has access to. You can specify this parameter multiple times, once for each Application. If this parameter is not specified, the user will have access to all Applications. These IDs must be the same as the primary key in the CleanSpeak Database. Here is an example of specifying multiple applications: <code>applicationID=1&amp;applicationID=2</code>	Integer	No

### Render JSON Response

This JSON response contains the information about the user. This returned from the GET and POST requests. The reason this is used for POST is to provide the ID of the user back to the caller.

```

{
  "user": {
    "id": 42,
    "email": "example@test.com",
    "roles": [
      {"id": 3, "name": "filter"},
      {"id": 5, "name": "moderator"}
    ]
    "applications": [
      {
        "id": 1,
        "name": "My Game",
        "components": [
          {"id": 1, "name": "Chat"},
          {"id": 2, "name": "Usernames"}
        ]
      },
      {
        "id": 2,
        "name": "My Website",
        "components": [
          {"id": 3, "name": "Profile Information"},
          {"id": 4, "name": "Forums"}
        ]
      }
    ]
  }
}

```

The JSON keys and values are described in this table:

Key	Value Description
user	The value for this key is a JSON object where the keys are the user fields and the values for each field.
user.id	The value for this key is an integer containing the user's ID.
user.email	The value for this key is a String containing the user's email or username.
user.roles	The value for this key is a JSON array that contains multiple JSON objects, one for each role that the user has.
user.roles.id	The ID of the role in the CleanSpeak Database.
user.roles.name	The name of the role in the CleanSpeak Database.
user.applications	The value for this key is a JSON array that contains multiple JSON objects, one for each Application that the user is assigned to.
user.applications.id	The ID of the Application in the CleanSpeak Database.
user.applications.name	The name of the Application in the CleanSpeak Database.
user.applications.components	This contains an array of JSON objects, one for each of the components within the Application.
user.applications.components.id	This ID of the Component in the CleanSpeak Database.
user.applications.components.name	This name of the Component in the CleanSpeak Database.

## Render XML Response

This XML response is nearly identical to the JSON response. The main difference is that the element names are not plural. Also, note that the XML uses the namespace <http://www.inversoft.com/schemas/cleanspeak/system/user/2>. Here is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <user id="42" email="test@example.com">
    <role id="3" name="filter"/>
    <role id="5" name="moderator"/>
    <application id="1" name="My Game">
      <component id="1" name="Chat"/>
      <component id="2" name="Username"/>
    </application>
    <application id="2" name="My Website">
      <component id="3" name="Profile Information"/>
      <component id="4" name="Forums"/>
    </application>
  </user>
</response>
```

## Authentication

### Webservices – Authentication

If you are using the Installable version of CleanSpeak and have enabled WebService authentication through the Management Interface, you will need to pass in the authentication key you setup in the Management Interface using the HTTP header named **Authentication** to the WebService.

Here is a Java sample that illustrates how this header is passed to the WebService:

```
URL url = new URL("http://localhost:8001/example");
URLConnection urlConnection = (URLConnection) url.openConnection();
urlConnection.setRequestProperty("Authentication",
    "your-authentication-key");
...
```

## Standard Error Response

### Webservices – Standard Error Response

All of the WebServices inside CleanSpeak use standard response messages and codes when errors occur. This covers the error codes and the response message format.

#### Response codes

You can determine if the request was successful or not based on the response code. Here are the response codes and their meanings:

Code	Description
200	The request was successful.
400	The request as invalid and malformed. The response will contain a JSON or XML message with the specific errors
401	WebService authentication is enabled and you did not supply a valid Authentication header. The response will be empty

### JSON Response

The error response contains two types of error messages: parameter errors and overall errors. Parameter errors occur when a parameter is invalid. All other errors are part of the overall error list. Here is an example of an error response:

```
{
  "parameterErrors": {
    "user.password": ["The user must have a password in the parameter [user.password]"],
    "user.email": ["The user must have an email (or username) in the parameter [user.email]"],
    "roles": ["You must specify at least one role for the user via the roles parameter"]
  },
  "errors": ["Unable to create the User"]
}
```

The top level JSON keys and values are described in this table:

Key	Value Description
parameterErrors	This key contains errors for any of the request parameters. The value is a JSON Object that contains individual keys for each parameter that has an error. The errors are stored in an array of Strings.
errors	This key contains the general errors that occurred during the request. The value is a JSON array that contains one or more Strings. Each String is a separate error message.

### XML Response

The error response in XML is primarily the same as the JSON response. Here is an example of the XML error response:

```
<?xml version="1.0" encoding="UTF-8"?>
<errors xmlns="http://www.inversoft.com/schemas/cleanspeak/error/2">
  <parameterError parameter="user.password" message="The user must have a password in the parameter [user.password]"/>
  <parameterError parameter="user.email" message="The user must have an email (or username) in the parameter [user.email]"/>
  <parameterError parameter="roles" message="You must specify at least one role for the user via the roles parameter"/>
  <error message="Unable to create the User"/>
</errors>
```

Here is a description of the elements under the **errors** root element:

Key	Value Description
parameterError	This element specifies an error for a request parameter. The name of the parameter with the error is specified by the parameter attribute and the error message is in the message attribute. There can be multiple parameterErrors elements for the same parameter.
error	This element specifies a general error. The message attribute contains the error message. There can be multiple error elements.

### Examples

## WebServices – Examples

This document covers how to quickly get started using the CleanSpeak Real-Time Filter WebService.

To get started using the WebService, you will need to send an HTTP request with the correct parameters. We'll cover two simple requests to the Real-Time Filter WebService using Java and PHP.

The Java examples only use classes from the JDK. We recommend using libraries for connecting to the WebService rather than JDK classes to help simplify your development. Here are some suggested libraries:

- [Apache HTTP Component Client](#) for making the HTTP request
- [JDOM](#) for parsing the XML
- [DOM4J](#) for parsing the XML
- [JSON Simple](#) for parsing the JSON

### RESTful Examples

Here are some examples of accessing a RESTful WebService using each of the above methods. These examples are all in Java, but other languages have similar APIs.

#### GET Example

```
URL url = new URL("http://localhost:8001/example/1");
HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
urlConnection.setDoInput(true);
urlConnection.setRequestMethod("GET");
urlConnection.connect();
```

#### POST Example

```
URL url = new URL("http://localhost:8001/example");
HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
urlConnection.setDoInput(true);
urlConnection.setDoOutput(true);
urlConnection.setRequestMethod("POST");
urlConnection.addRequestProperty("Content-Type",
"application/x-www-form-urlencoded");
urlConnection.connect();

OutputStream out = urlConnection.getOutputStream();
OutputStreamWriter writer = new OutputStreamWriter(out);
writer.append("someParameter=").append(URLEncoder.encode("Parameter value",
"UTF-8")).append("&");
writer.append("anotherParameter=").append(URLEncoder.encode("Some other
value", "UTF-8"));
writer.flush();
```

#### PUT Example

```
URL url = new URL("http://localhost:8001/example/1");
URLConnection urlConnection = (URLConnection) url.openConnection();
urlConnection.setDoInput(true);
urlConnection.setDoOutput(true);
urlConnection.setRequestMethod("PUT");
urlConnection.addRequestProperty("Content-Type",
    "application/x-www-form-urlencoded");
urlConnection.connect();

OutputStream out = urlConnection.getOutputStream();
OutputStreamWriter writer = new OutputStreamWriter(out);
writer.append("someParameter=").append(URLEncoder.encode("Parameter value",
    "UTF-8")).append("&");
writer.append("anotherParameter=").append(URLEncoder.encode("Some other
value", "UTF-8"));
writer.flush();
```

#### **DELETE Example**

```
URL url = new URL("http://localhost:8001/system/user.js?id=1234");
URLConnection urlConnection = (URLConnection) url.openConnection();
urlConnection.setDoInput(true);
urlConnection.setDoOutput(true);
urlConnection.setRequestMethod("DELETE");
urlConnection.connect();

if (urlConnection.getResponseCode() == 200) {
    // Success
} else {
    // Error
}
```

#### **Java Locate Operation (using XML)**

This example code illustrates a locate operation in Java that returns XML results.

```

URL url = new URL("http://localhost:8001/content/item.xml");
URLConnection urlConnection = (URLConnection) url.openConnection();
urlConnection.setDoInput(true);
urlConnection.setDoOutput(true);
urlConnection.setRequestMethod("POST");
urlConnection.addRequestProperty("Content-Type",
"application/x-www-form-urlencoded");
// If you Webservice authentication turned on
// urlConnection.addRequestProperty("Authentication",
"your-authentication-key");
urlConnection.connect();

OutputStream out = urlConnection.getOutputStream();
OutputStreamWriter writer = new OutputStreamWriter(out);
writer.append("contentItem.content=").append(URLEncoder.encode("Some
content to filter", "UTF-8")).append("&");
writer.append("contentItem.type=text&");
writer.append("filter.operation=locate&");
writer.append("filter.blacklist.enabled=true");
writer.flush();
writer.close();

InputStream inputStream = urlConnection.getInputStream();
Document dom =
DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(inputStream);
Element filter = (Element)
dom.getDocumentElement().getElementsByTagName("filter").item(0);
NodeList matches = filter.getElementsByTagName("match");
for (int i = 0; i < matches.getLength(); i++) {
    Element match = (Element) matches.item(i);
    System.out.println("Found match at " + match.getAttribute("start") + "
" + match.getAttribute("length"));
}
inputStream.close();

```

### **Java Replace Operation (using XML)**

This example code illustrates a replace operation in Java that returns XML results.

```

URL url = new URL("http://localhost:8001/content/item.xml");
URLConnection urlConnection = (URLConnection) url.openConnection();
urlConnection.setDoInput(true);
urlConnection.setDoOutput(true);
urlConnection.setRequestMethod("POST");
urlConnection.addRequestProperty("Content-Type",
"application/x-www-form-urlencoded");
// If you have Webservice authentication turned on
// urlConnection.addRequestProperty("Authentication",
"your-authentication-key");
urlConnection.connect();

OutputStream out = urlConnection.getOutputStream();
OutputStreamWriter writer = new OutputStreamWriter(out);
writer.append("contentItem.content=").append(URLEncoder.encode("Some
content to filter", "UTF-8")).append("&");
writer.append("contentItem.type=text&");
writer.append("filter.operation=replace&");
writer.append("filter.blacklist.enabled=true");
writer.flush();
writer.close();

InputStream inputStream = urlConnection.getInputStream();
Document dom =
DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(inputStream);
Element filter = (Element)
dom.getDocumentElement().getElementsByTagName("filter").item(0);
Element replacement = (Element)
filter.getElementsByTagName("replacement").item(0);
System.out.println("Replacement is " + replacement.getTextContent());
inputStream.close();

```

### ***PHP Locate Operation (using XML)***

This example code illustrates a locate operation in PHP that returns XML results.



```

$data = "contentItem.content=" . urlencode("Some content to filter") . "&"
.
  "contentItem.type=text&" .
  "filter.operation=locate&" .
  "filter.blacklist.enabled=true";

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, "http://localhost:8001/content/item.xml");
curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type:
application/x-www-form-urlencoded'));
// If you have Webservice authentication turned on
// curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type:
application/x-www-form-urlencoded', 'Authentication:
your-authentication-key'));
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $data);

$response = curl_exec($ch);
curl_close($ch);

$parser = xml_parser_create();
xml_parser_set_option($parser, XML_OPTION_CASE_FOLDING, 0);
xml_parse_into_struct($parser, trim($response), $structure, $index);
xml_parser_free($parser);
foreach ($structure as $s) {
    if ($s['tag'] == "match") {
        print("Found match at " . $s['attributes']['start'] . " " .
        $s['attributes']['length'] . "\n");
    }
}

```

### ***PHP Replace Operation (using XML)***

This example code illustrates a replace operation in PHP that returns XML results.

```

$data = "contentItem.content=" . urlencode("Some content to filter") . "&"
.
"contentItem.type=text&" .
"filter.operation=replace&" .
"filter.blacklist.enabled=true";

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, "http://localhost:8001/content/item.xml");
curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type:
application/x-www-form-urlencoded'));
// If you have Webservice authentication turned on
// curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type:
application/x-www-form-urlencoded', 'Authentication:
your-authentication-key'));
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $data);

$response = curl_exec($ch);
curl_close($ch);

$parser = xml_parser_create();
xml_parser_set_option($parser, XML_OPTION_CASE_FOLDING, 0);
xml_parse_into_struct($parser, trim($response), $structure, $index);
xml_parser_free($parser);
foreach ($structure as $s) {
    if ($s['tag'] == "replacement") {
        print("Replacement is " . $s['value'] . "\n");
    }
}

```

## Moderation Notifications

The CleanSpeak Moderation Notification system is designed to provide feedback to your system when moderators perform specific actions on content or users. Currently, CleanSpeak supports notifications for the following:

- Approvals
- User Actions
- Content Edits
- Content Deletes

### Approvals

The Moderator Approval Queue supports the approval and rejection of content. Moderators login to the CleanSpeak Management Interface and from there can approve and reject content. Once moderators have approved or rejected content, CleanSpeak will send notifications back into your system.

### HTTP Request Specification

The approval notification HTTP request specification

Name	Description	Type
------	-------------	------

type	This parameter specifies the type of notification  Example: <code>type=moderation&amp;</code>	String
moderation.item[index].uid	This parameter specifies the UID of the content that was moderated. The index part will be a integer value that starts at 0 and increments for each piece of content that was moderated. This allows the results for multiple pieces of content to be notified in a single request. Here is an example of a notification for multiple pieces of content:  Example: <code>moderation.item[0].uid=12345&amp;</code> <code>moderation.item[1].uid=67890&amp;</code>	String
moderation.item[index].action	This parameter specifies the action that was taken on the content. The possible values are <b>approved</b> and <b>rejected</b> . The index part will be a integer value that starts at 0 and increments for each piece of content that was moderated. This allows the results for multiple pieces of content to be notified in a single request. Here is an example of a notification for multiple pieces of content:  Example: <code>moderation.item[0].action=approved&amp;</code> <code>moderation.item[1].action=rejected</code>	String

## User Actions

The User Action system provides moderators the capability to perform specific actions on users that generate content. Actions such as ban, mute, and kick are a few examples of actions that moderators can take when, for instance, users are generating inappropriate content. When actions are taken, CleanSpeak will send notifications back into your system.

## HTTP Request Specification

The user action notification HTTP request specification

Name	Description	Type
type	This parameter specifies the type of notification  Example: <code>type=userAction&amp;</code>	String
userAction.userUID	This parameter specifies the unique identifier of the user the action is being performed. This UID is equivalent to the 'contentItem.senderUID' you provide when moderating and storing content.  Example: <code>userAction.userUID=JoeBlow&amp;</code>	String
userAction.application	This parameter specifies the application that the action is being performed in  Example: <code>userAction.application=MyApp&amp;</code>	String
userAction.component	This parameter specifies the component that the action is being performed in  Example: <code>userAction.component=MyComponent&amp;</code>	String

userAction.action	This parameter specifies the name of the action that's occurring  Example: <code>userAction.action=mute&amp;</code>	String
userAction.key	This parameter specifies the name of the key that's occurring for the given action  Example: <code>userAction.key=30secs</code>	String

## Content Edits

Depending on your Component configuration, content within CleanSpeak can be edited by moderators. When content is edited, CleanSpeak will notify your system so that your system can process the edit appropriately.

### HTTP Request Specification

The Content Edit notification HTTP request specification

Name	Description	Type
type	This parameter specifies the type of notification  Example: <code>type=contentItemEdit</code>	String
contentItem.uid	This parameter specifies the unique identifier of the content being edited. This UID is equivalent to the 'contentItem.uid' your system generates and provides when moderating and storing content.  Example: <code>contentItem.uid=1</code>	String
contentItem.application	This parameter specifies the application the content was generated in:  Example: <code>contentItem.application=MyApp</code>	String
contentItem.component	This parameter specifies the component that the content was generated in;  Example: <code>contentItem.component=MyComponent&amp;</code>	String
contentItem.content	This parameter specifies the content that the moderator added as a result of the edit:  Example <code>contentItem.content=Edited%20by%20moderator</code>	

## Content Deletes

Depending on your Component configuration, content within CleanSpeak can be deleted by moderators. When content is deleted, CleanSpeak will notify your system so that your system can process the delete appropriately.

### HTTP Request Specification

The Content Delete notification HTTP request specification

Name	Description	Type
type	This parameter specifies the type of notification  Example: <code>type=contentItemDelete</code>	String
contentItem.uid	This parameter specifies the unique identifier of the content being edited. This UID is equivalent to the 'contentItem.uid' your system generates and provides when moderating and storing content.  Example: <code>contentItem.uid=1</code>	String
contentItem.application	This parameter specifies the application the content was generated in:  Example: <code>contentItem.application=MyApp</code>	String
contentItem.component	This parameter specifies the component that the content was generated in;  Example: <code>contentItem.component=MyComponent&amp;</code>	String

**Next Step:** [Notification Request Handler](#)

## Notification Request Handler

### The Notification Request Handler

In order to appropriately handle requests from the notification system, you must build a simple HTTP request handler that listens for requests from the CleanSpeak Moderation Notification system. Your request handler must be designed to respond to simple HTTP POST requests using the request specifications provided in the [Moderation Notifications](#) section.

### Responses

Your notification WebService must handle the RESTful request described above and send back an appropriate response. Your notification WebService must send back to CleanSpeak an HTTP response code that indicates whether or not the notification was successfully handled or not. If your WebService handled the notification properly, it must send back a HTTP response status code of 200. If there was any type of error or failure, your WebService must send back a HTTP response status code of 500.

### Configuration

Once your web service request handler is complete and listening for notification requests, you must inform CleanSpeak of your web service IP or domain name. Notification servers can be added by going to the CleanSpeak Management Interface 'System > Servers' interface and configuring the IP/domain name in addition to the 'notification' server type. CleanSpeak will round-robin multiple notification servers if more than one is desired.

Once CleanSpeak knows about your notification server, you must then enable either approvals and/or user actions in the CleanSpeak Management Interface 'System > Applications' interface for the particular application-component combination desired.

## Updates and Patches

### CleanSpeak Updates and Patches

Periodically, Inversoft releases updates for the CleanSpeak. You might also require a patch if you are running into problems and have contacted Inversoft Support for assistance. In these cases, you will need to update the application.

### Linux

Updating your application is easy if you installed using the RPM or Debian packages. All you need to do is to issue an update command to the dpkg or rpm program and specify the new package file. Here is an example:

```
# For the Management Interface on Debian
$ dpkg -i cleanspeak-management-interface-<version>.deb

# For the Management Interface on Redhat
$ rpm -U cleanspeak-management-interface-<version>.rpm
```

## Windows

On Windows, the steps required are different. Follow these steps to upgrade your version on Windows:

1. Download the webapp only bundle
2. Shut-down your Tomcat or your JEE server
3. Rename the CLEANSPEAK\_HOME/web directory to something meaningful like web-2009-06-07 (this is helpful in case you need to revert to this version of the application)
4. Extract the update bundle to the CLEANSPEAK\_HOME directory. This will place a new directory called web in the directory
5. Restart Tomcat

## Database

Depending on your current version and the new version you will be updating to you might need to execute one or more SQL scripts to update your database. These scripts are listed on the [Downloads](#) page under the Upgrade Downloads section. To determine which scripts you might need to run, determine the current version you are running. Based on that version number, run all of the scripts where the first version is your version number or higher. Continue running each script until you reach a script where the second number is higher than the version you are updating to.

Here is the types of versions that CleanSpeak has:

1. 1.0 is a major version
2. 1.2 is a minor version
3. 1.2.1 is a patch version
4. 1.4-RC1 is a preview version

The ordering of the versions is as follows:

1. Major Preview Version
2. Major Version
3. Patch Version(s)
4. Minor Preview Version
5. Minor Version
6. Patch Version(s)

For example, here are a number of versions in order:

1. 1.0-RC1
2. 1.0-RC2
3. 1.0-RC3
4. 1.0
5. 1.0.1
6. 1.0.2
7. 1.1-RC1
8. 1.1-RC2
9. 1.1-RC3
10. 1.1-RC4
11. 1.1
12. 1.1.1
13. 1.1.2
14. 1.1.3
15. 1.1.4

## Migrating Between Environments

In some cases, you might have multiple installations of CleanSpeak in different environments and migrate between environments. Here's a common example:

1. Update CleanSpeak from 1.0.1 to 1.0.2 in your staging environment
2. Edit various white and black lists using the Management Interface in your staging environment

3. Test your changes inside your game or application in your staging environment
4. Update CleanSpeak from 1.0.1 to 1.0.2 in your production environment
5. Migrate your database changes from your staging to production environment

These steps are all very straight forward, except for the last step. This step can be difficult if you are using both the Real-Time Filter and the Moderator products. The reason is that you don't want to migrate the entire staging database from staging to production, because this will end up clobbering all of the content handled by the Moderator product. Instead, you need to do a partial migration of the database between environments.

The process for a partial migration of the database for the Real-Time Filter is as follows:

1. Backup your production CleanSpeak database entirely
2. Copy these tables from your staging environment to your production environment in order:
  - tags
  - blacklist\_entry\_modifications
  - blacklist\_entry\_modifications\_tags
  - whitelist\_entry\_modifications
  - blacklist\_entries
  - blacklist\_entries\_tags
  - whitelist\_entries
  - verification\_cases

## Upgrading from 1.x

### Upgrading from 1.x

If you are upgrading from any version 1.x you will need to take a number of steps to complete the upgrade. First, review the [Updates and Patches](#) page, then follow these steps:

#### Pending Approvals

First, login to your existing Management Interface and check to see if there are any pending approvals *System -> Approvals*. Approve or reject any that are pending. There must be no pending approvals prior to the upgrade.

#### Database Migration

The core change related to filtering with CleanSpeak 2.0 is that the concept of Contexts has been removed and instead of having one Category, each blacklist entry can now have multiple Tags. Your existing Contexts and Categories will both be migrated as Tags for each entry. Check the version you are currently running. You must be at 1.3.x in order to migrate to 2.0. If you are running 1.0, 1.1, or 1.2, you need to run the respective migration scripts to get to 1.3 first. All update scripts are available in your [account](#) from *Downloads -> Upgrade Downloads*. *PostgreSQL Users*: You will need to ensure that the user you run the update as has permissions to create store procedures and also has permissions to load C based stored procedures from disk. You also need to ensure that you have the UUID-OSSP library installed in your PostgreSQL installation because it is loaded as part of the upgrade process. To give the **cleanspeak** user these permissions, you can grant the SUPERUSER role to it and then later revoke it. The steps to grant this role are:

```
$ psql -Upostgres cleanspeak
postgres> alter role cleanspeak SUPERUSER
```

#### Software Update

CleanSpeak 2.0 downloads are available from your [account](#). If you will be using the [Moderation Tools](#), you will need to install the Search Engine after updating the Management Interface and Webservice.

#### Tomcat Updates

You must reconfigure the Tomcat server.xml using the server.xml-example files in the distributions. This needs to be done for the Management Interface and WebService. We updated to Tomcat 7.0 and this new version of Tomcat uses different server.xml configuration parameters.

#### Integration Code

CleanSpeak 1.0 implemented an XML API and later introduced a REST API with version 1.2. The XML API has been completely removed with 2.0 and the REST API has been changed. Review the [WebServices](#) section and specifically [Content Items](#) to update your filtering integration code.

#### 1.x Moderator

If you are using the 1.x Moderator product, [contact us](#) for details with the upgrade.

## License File

Your 1.x license file will need to be updated. Login to your [account](#) and click *Generate 2.x License File* and review the [license file installation doc](#).

## Changes to REST API – Quick Reference

Use the table below to identify the old parameter name and the new parameter name. New Parameters that are bold indicate that they are required. For XML over HTTP integrations, [contact us](#) with any REST implementation questions.

Old Parameter	New Parameter
request.content	<b>contentItem.content</b>
*request.type	<b>contentItem.type</b>
request.filter.enabled request.filter.cleanspeakdb.enabled	<b>filter.blacklist.enabled</b>
request.filter.operation	<b>filter.operation</b>
request.filter.characters.enabled	filter.characters.enabled
request.filter.characters.character	filter.characters.character
**request.filter.cleanspeakdb.category	filter.blacklist.tag
request.filter.cleanspeakdb.locale	filter.blacklist.locale
request.filter.cleanspeakdb.severity	filter.blacklist.severity
request.filter.emails.enabled	filter.emails.enabled
request.filter.phoneNumbers.enabled	filter.phoneNumbers.enabled
request.filter.urls.enabled	filter.urls.enabled
request.filter.words.enabled	filter.words.enabled

\*This was not a required parameter with 1.x but is now required with 2.0

\*\*Categories (and Contexts) have been changed to Tags.

## Upgrading from 2.0 Pre-Releases

### Upgrading from 2.0 Pre-Releases

The migration from any pre-release of 2.0 (2.0-Bx or 2.0-RCx) to the full version of 2.0 requires a number of modifications to work properly. These changes are outlined below.

#### 1. Migrate your database from your existing pre-release version to 2.0

A database migration is required to upgrade your database from your pre-release version to 2.0. Please contact us at [support@inversoft.com](mailto:support@inversoft.com) to obtain the scripts necessary to upgrade your database to the latest version.

#### 2. Update your license file to a 2.0 license

You can download a 2.0 license file from the Inversoft website and install it as normal.

#### 3. Change your API calls to the new API URLs and parameters

To increase performance and provide consistency across all our APIs, we have made a number of minor changes to the APIs. To make the update as simple as possible, follow the instructions below to update your application code to use the new APIs.

#### 4. Reindex your search index

If you've licensed the Moderation tools, after 2.0 is up-and-running, you'll need to re-index your search index. You can accomplish this by logging in as an admin going to 'System > Search Index'.

### Changes to /content/handle



- /content/handle has been moved to /content/item
- The names of the request parameters for the /content/item request have changed. Rather than prefixing everything with “request.” we have changed the request parameters to indicate their use. Use the table below to identify the old parameter name and the new parameter name. NOTE: the values you need to pass in the request have NOT changed

The changes to the request parameters from /content/handle to /content/item are listed in this table:

Old Parameter	New Parameter
request.content	contentItem.content
request.contentType	contentItem.type
request.application	contentItem.application
request.component	contentItem.component
request.moderation.createDate	contentItem.createInstant
request.moderation.locationUID	contentItem.location
request.moderation.senderUID	contentItem.senderUID
request.moderation.receiverUID	contentItem.receiverUID
request.moderation.clientUID	contentItem.uid
request.moderation.type	moderation.type
request.filter.operation	filter.operation
request.filter.cleanspeakdb.enabled	filter.blacklist.enabled
request.filter.cleanspeakdb.tag	filter.blacklist.tag
request.filter.cleanspeakdb.locale	filter.blacklist.locale
request.filter.cleanspeakdb.severity	filter.blacklist.severity
request.filter.characters.enabled	filter.characters.enabled
request.filter.characters.character	filter.characters.character
request.filter.emails.enabled	filter.emails.enabled
request.filter.emails.maximumMatchLength	filter.emails.maximumMatchLength
request.filter.emails.spacePenalty	filter.emails.spacePenalty
request.filter.emails.domainQuality.domain	filter.emails.domainQuality.domain
request.filter.emails.domainQuality.quality	filter.emails.domainQuality.quality
request.filter.phoneNumbers.enabled	filter.phoneNumbers.enabled
request.filter.phoneNumbers.maximumMatchLength	filter.phoneNumbers.maximumMatchLength
request.filter.phoneNumbers.minimumMatchLength	filter.phoneNumbers.minimumMatchLength
request.filter.phoneNumbers.spacePenalty	filter.phoneNumbers.spacePenalty
request.filter.phoneNumbers.separatorPenalty	filter.phoneNumbers.separatorPenalty
request.filter.phoneNumbers.wordPenalty	filter.phoneNumbers.wordPenalty
request.filter.urls.enabled	filter.urls.enabled
request.filter.urls.maximumMatchLength	filter.urls.maximumMatchLength
request.filter.urls.spacePenalty	filter.urls.spacePenalty
request.filter.urls.domainQuality.domain	filter.urls.domainQuality.domain
request.filter.urls.domainQuality.quality	filter.urls.domainQuality.quality

request.filter.words.enabled	filter.words.enabled
request.filter.words.word	filter.words.word

#### Changes to /content/user

- The request parameters for the /content/user request have changed names and in some cases the values you need to send in are different
- The HTTP methods used to call the /content/user have changed. No longer is the POST method used for creates and updates. Instead, the POST method now only performs a create while the PUT method is used for updates.

The changes to the request parameters for /content/user are listed in this table:

Old Parameter	New Parameter
uid	contentUser.uid
name	contentUser.name
email	contentUser.email
birthDate	contentUser.birthDateInstant
lastLogin	contentUser.lastLoginInstant
attributes['name']	contentUser.attributes['name']

## Troubleshooting

### CleanSpeak Troubleshooting

If any problems arise or if you are unable to access the Webservice or the Management Interface, consult the log files located in the Tomcat installation directory under the CATALINA\_HOME/logs directory. In most cases all errors will be written out to the catalina.out log file. You can contact Inversoft support to help troubleshoot any possible errors that might exist in this file.

## User Guide

Welcome to the CleanSpeak 2.0 User Guide. This guide is for moderation managers, community managers, project managers, and moderators to:

1. Align your CleanSpeak configuration with your business requirements.
2. Learn how to use the CleanSpeak management interface for moderation and filter management on a regular basis.

For technical information, refer to the [Technical Guide](#).

#### Next Step: [Concepts](#)

Search this guide:

- [Concepts](#)
- [First Login](#)
- [List Management](#)
- [Moderation](#)
- [FAQ & Troubleshooting](#)

## Concepts

### CleanSpeak – Overview

There are two general methods to manage user-generated content: Automated filtering and human moderation. The CleanSpeak platform allows you to marry the two concepts so that they compliment each other according to your business needs.

As you read through this guide, consider your audience and goals for your community to determine how much to leverage automation versus human involvement.

- [Filtering Concepts](#)

- [Moderation Concepts](#)

## Filtering Concepts

Applying the CleanSpeak real-time filter may include one or more of the following categories: Blacklist filtering, whitelist filtering, and graylist filtering.

### Blacklist Filtering

Blacklist filtering prevents words/phrases from being used. Possible outcomes when CleanSpeak finds a match (or matches) on the blacklist include:

- Reject the entire message
- Replace the match(es) with a character or word (\*\*\*\*)
- Display the message to the author only but not display it to the world

You may also take additional actions within your application based on the type of match CleanSpeak finds depending on your business requirements, such as:

- Allow the user to resubmit the content in the case of forum posts, comments, and reviews
- Send a warning message to the user
- Silence or kick the user from the application

All of the above actions can be customized by configuring black list entries with severity levels and categories (Tags). Visit the [List Management](#) section of this guide for details.

### Whitelist Filtering

Whitelist filtering is a method of checking words as they are typed to allow only words on a pre-approved list. When whitelist filtering, a blacklist of phrases is also used as a final filtering check when the user submits a message. For example, the words "hard" and "on" may be allowed individually, but the phrase "hard on" would be on the black list.

### Graylist Filtering

Graylist filtering is a method of alerting human moderators when a word or phrase is used. Typically, when the word/phrase is found, the message is allowed to be displayed, but you may also decide to reject the match as described in Blacklist Filtering above.

The CleanSpeak moderation platform implements graylist filtering through the use of [Alert Queues](#).

### Which method should you use?

Filtering techniques can be implemented individually or all applied at the same time! Consider the case of white list filtering with a gray list of phrases to be warned about while black listing phrases of white listed words. Sound confusing? Here's an example for a children's application: "home" and "mom" may be on the whitelist, but when used in a sentence like "is your mom home?", the content can be blocked (blacklist) and the user reported immediately as a potential predator (graylist).

For assistance integrating the filter with your business requirements, contact us at [support@inversoft.com](mailto:support@inversoft.com).

**Next Step:** [Moderation Concepts](#)

## Moderation Concepts

CleanSpeak allows moderators to prioritize content to be viewed through the use of content queues, search for content in real-time with a comprehensive search form, review details of users and their content history, take actions on users, and more. This section describes the different types of queues and how they can be implemented.

### Pending Approval Queue

Content may be placed into a pending state waiting for a human moderator to approve or reject the content, such as profile information or images. A message is typically sent to the user upon submission that the content will be displayed after a moderator has reviewed it.

### Alert Queue

The alert queue is populated with high priority content through the use of graylist filtering as described in the [Filtering Concepts](#) section. Alerts are generated when the filter finds a match on a specific Tag or Tags as configured in the [Setup & Configuration](#) section. Possible applications include being alerted when your users discuss your competition, when they exhibit grooming or predatory behaviors, or attempt to share personally identifiable information (PII).

## Pending Review Queue

When implementing the review queue, every single piece of content sent to CleanSpeak will be marked as "pending review" and sorted in the queue with higher priority content at the top of the list to be reviewed first. Possible applications include under-13 chat where all content needs to be reviewed or forum posts that should all be read by a moderator.

## Which queue(s) should you use?

CleanSpeak can be configured through the use of Applications & Components to use different queues for various content sources. For example, you may have a forum as well as real-time chat in your application. You may elect to use the approval or review queue for the forum and configure chat to generate alerts within an alert queue.

For assistance integrating queues with your business requirements, contact us at [support@inversoft.com](mailto:support@inversoft.com).

**Next Step:** [First Login](#)

## First Login

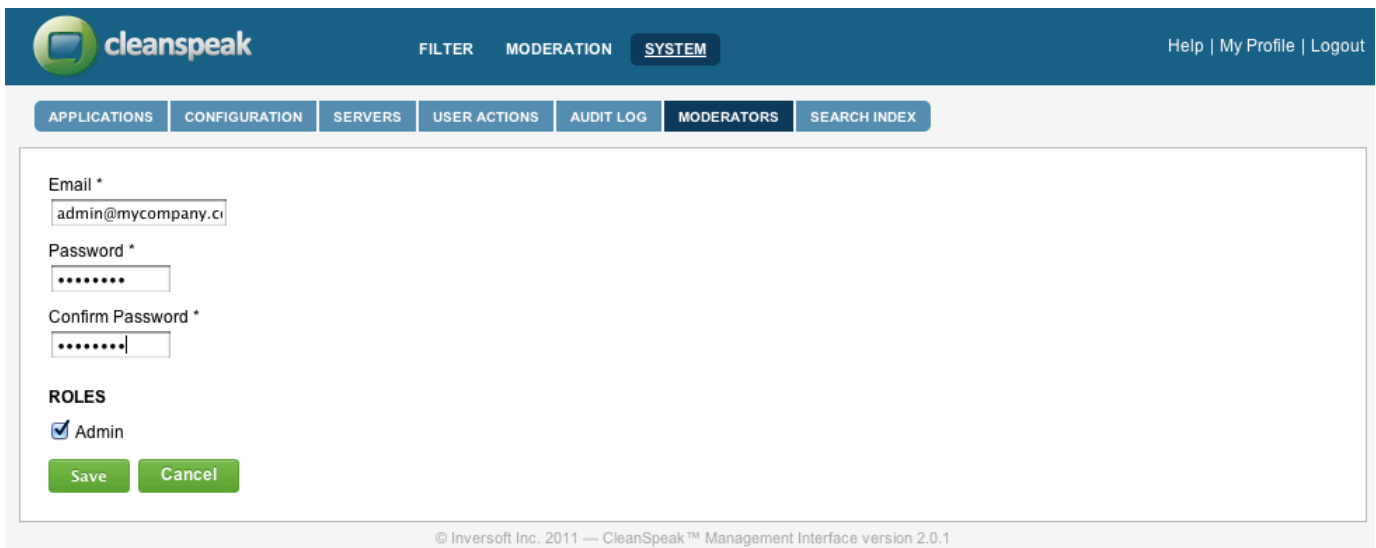
You should have a URL address to access the CleanSpeak management interface. If not, contact your IT department for access.

The root administrator access for CleanSpeak after a new install:

**Email:** [admin@inversoft.com](mailto:admin@inversoft.com)

**Password:** password

Upon initial login, it is recommended to create a new administrator account via *System -> Moderators -> Add Moderator*. Check the box "Admin" under Roles. Remove the original admin account after logging into your newly created account.



The screenshot displays the CleanSpeak Management Interface. The top navigation bar includes the CleanSpeak logo, menu items for FILTER, MODERATION, and SYSTEM (which is highlighted), and links for Help, My Profile, and Logout. Below the navigation bar is a secondary menu with buttons for APPLICATIONS, CONFIGURATION, SERVERS, USER ACTIONS, AUDIT LOG, MODERATORS (which is selected), and SEARCH INDEX. The main content area shows a form for adding a moderator. It includes fields for Email (with the value admin@mycompany.ci), Password, and Confirm Password. Below these fields is a section titled ROLES with a checked checkbox for Admin. At the bottom of the form are Save and Cancel buttons. A footer at the bottom of the page reads: © Inversoft Inc. 2011 — CleanSpeak™ Management Interface version 2.0.1

The management interface has three primary sections:

### Filter

- Add, edit, and delete entries from your lists
- Categorize entries with Tags to be used to generate alerts and customize the behavior of the filter
- Verification tool to test the accuracy of the filter
- Review changes to your lists with the Approval system prior to them taking effect
- Import/Export lists used with filtering

### Moderation

- Search on content and users
- Take action on content pending approval
- Manage alerts generated by the filter
- Review the history and patterns of individual users
- Take direct actions on users

- Review event logs for users

## System

- Set global configuration parameters
- Manage moderator accounts

**Next Step:** [Setup System Users](#)

## System Users & Permissions

Roles may be assigned for each moderator that has access to CleanSpeak. When the “Admin” box is checked, the other roles will be hidden. When assigning Filter and Moderator roles, the moderator can be restricted to particular application(s). If no applications are selected, then the moderator will have access to all applications.

Filter Roles	Moderation Roles	Other Roles
<input checked="" type="radio"/> Filter <input type="radio"/> Filter Manager <input type="radio"/> None	<input type="radio"/> Moderator <input checked="" type="radio"/> Moderation Manager <input type="radio"/> None	<input checked="" type="checkbox"/> View Audit Logs <input checked="" type="checkbox"/> Administer Users

### Filter Roles

- **Filter:** Modify the black and whitelists. Use the verification tool to test the filter. Cannot approve list changes.
- **Filter Manger:** All of “Filter” role abilities. Can also approve changes made to the filter lists, edit tags, and import/export lists

### Moderator Roles

- **Moderator:** Action content from queues, escalate content to a manager from any queue, perform actions on users, comment on users.
- **Moderator Manager:** All of “Moderator” role abilities. Can also view and take action on escalations.

### Other Roles

- **View Audit Logs:** View records from *System -> Audit Log*
- **Administer Users:** Add, edit, delete moderator accounts from *System -> Moderators*

**Next Step:** [List Management](#)

## List Management

CleanSpeak uses the blacklist (*Filter -> Blacklist*) for both graylist and blacklist filtering as described in the [Filtering Concepts](#) section. Each entry in the blacklist has many configuration options to enable the filter to perform different types of actions. This page provides an outline to apply to your business requirements with severity levels and tags. You can jump to detailed list management pages with the links below.

- [Add/Edit Blacklist Entries](#)
- [Testing the Filter](#)
- [Approving Changes](#)
- [Whitelist](#)
- [Import/Export Lists](#)
- [Multilingual](#)

## Severity

There are five different severity levels: Severe, High, Medium, Mild, and None. The blacklist that is initially installed has been designed with the following filter actions in mind:

Severity	Recommended Action
Severe	Reject the entire post. Few entries are marked as Severe to ensure accuracy when rejecting content.
High	Youth Audience: Reject the entire post. Mature Audience: Reject post or replace match with asterisks.

Medium	Youth Audience: Display message to the author only but not to the public. Mature Audience: Replace with asterisks or allow unaltered message to be submitted.
Mild	Youth Audience: Display message to the author only but not to the public. Mature Audience: Allow unaltered message to be submitted.
None	Do not take any action to the content.

When using the CleanSpeak moderation features, severity level is also used to prioritize content that is displayed in Review and Alert queues.

## Tags

Tags are classifications of blacklist entries. The blacklist that is initially installed includes the following tags: Alcohol, AsciiArt, Bullying, Drug, Gang, Racial, Religious, Slang, Weapons, and Youth. An entry may have more than one tag.

When defining your filtering actions, you can take different actions based on Tags. Tags can be added, edited, or deleted via *Filter -> Tags*.

The moderation system generates alerts from Tags as described in the [Moderation Concepts](#) section. Individual components may be configured to generate different types of alerts. Visit the [Setup & Configuration](#) page for details.

## How to Apply Severity & Tags

Once you have determined the filter actions based on severity level and tag(s), inform your development team who can apply those actions by either [specifying attributes in the request](#), implementing different actions based on the [filter response](#), or a combination of both.

**Next Step:** [Add/Edit Blacklist Entries](#)

### Add/Edit Blacklist Entries

The add/edit screen is viewed either by clicking an existing blacklist entry or clicking the *Add Entry* button from *Filter -> Blacklist*. There are three sections to configure for each entry: Basics, Tags, and Misc. Once a change has been made, the entry will enter a state of [pending approval](#). Once Approved, the change will be effective in the live application.

[ [Basics](#) ] [ [Tags](#) ] [ [Miscellaneous](#) ] [ [Filter Modes](#) ]

### Basics

**Text (Required):** Base Entry. If the entry is a phrase, insert a hyphen "-" in place of the space(s), such as: bite-me or my-computer-crashed.

**Severity (Required):** Harshness of the entry set to none, mild, medium, high, or severe. Severity may be used to filter specific severity levels or to customize responses to filter matches. Severity is also used to prioritize content when matches are found to sort in queues. See the [list management overview](#) page for details.

**Locale (Required):** Language of the entry and optionally the country code. Supported formats are [ISO 639-1 two-digit language code](#) for the language followed by an optional underscore and [ISO 3166 two-digit country code](#).

Examples: **en** = English, **es** = Spanish, **en\_US** = American English, **es\_MX** = Mexican Spanish.

See the [Multilingual](#) section of this guide for details.

**Variations (Optional):** Field to add alternate spellings of root entries. Variations should only contain misspellings, inflections (see [Multilingual](#)), or alternate ways of communicating the base entry. If the variation is a phrase, insert a hyphen "-" in place of the space(s), such as: bite-me or my-computer-crashed. Multiple variations can be added with a single space between each entry. Note that the filter will look for inflections based on the selected parts of speech for variations as well as the base entry.

**Definition (Optional):** Field to reference why the entry was added that is not used by the filter in any way.

**Basics**

Text \*

Severity \*

Select ▾

Locale \*

Variations

Definition

## Tags

Categories of entries. An entry may have multiple tags but must have at least one. New tags can be created by using the text field and clicking the green + sign. Tags may also be managed via *Filter -> Tags*.

Tags are used to customize filter behavior by telling the filter which tags to filter against when sending content to CleanSpeak. They may also be used to build automated reactions when a match is found by checking the response from the filter. For example, if the match found has a tag name of Drug & Alcohol, you could display a warning message with a reminder that chatter about drugs and alcohol is not permitted. See the [list management overview](#) page for details.

The moderation system also uses tags to generate behavioral alerts (graylisting). Refer to the [Moderation Concepts](#) page.

**Tags**

Tags\*

 +  
 AsciiArt  
 Bullying  
 Competitors  
 Drug & Alcohol  
 Gang  
 Grooming  
 Profanity  
 Racial  
 Religious  
 Weapons  
 Youth

## Miscellaneous

**Parts of Speech:** Used to find inflections of root entries. Check the part(s) of speech if you want CleanSpeak to find inflections for you automatically based on the locale. Currently, CleanSpeak provides inflection support for English, Spanish, French, German, Italian, Dutch, Polish, and Russian. See the [Multilingual](#) page for details.

**Filter Mode:** Specifies how the filter should look for the blacklist entry. Details for each mode are described below.

**Show Combinations:** Displays a list of all inflections the filter will look for automatically from parts of speech, the base entry, variations, and locale.

**Misc.**

Adjective

Adverb

Noun

Verb

Filter Mode

## Filter Modes

The root entry used in the examples below is **fawk** configured as a **verb**.

Mode	Description	Match Examples	Non-Match Examples
<b>Exact Match</b>	Instructs the filter to find a match for the entry when the character string is an exact match of the entry. The filter will also match on variations, expansions, and is case insensitive.	<ul style="list-style-type: none"> <li>fawk</li> <li>FaWk</li> <li>fAwkiNg</li> </ul>	<ul style="list-style-type: none"> <li>f awk</li> <li>f@wk</li> <li>f.awk</li> </ul>
<b>Non-Embeddable</b>	Includes conditions for exact_match. Instructs the filter to look for repeat characters, replacement characters, leet speak, and added spaces and punctuation <i>unless</i> characters are in front of or behind the entry without spaces.	<ul style="list-style-type: none"> <li>faawwwwwk</li> <li>faVvk</li> <li>f.a.w.k</li> <li>f a w k</li> </ul>	<ul style="list-style-type: none"> <li>sfawk</li> <li>fawkface</li> <li>2fawk</li> </ul>
<b>Embeddable</b>	Includes conditions for exact_match and non_embeddable. Instructs the filter to find the entry in cases where numbers or words (that are on the whitelist) are in front of or behind the entry without spaces.	<ul style="list-style-type: none"> <li>faVvkhead</li> <li>bluefaawwwwwk</li> <li>f.@.w.k5323</li> </ul>	<ul style="list-style-type: none"> <li>fooblahfawk</li> <li>fawkzee</li> </ul>
<b>Distinguishable</b>	Includes conditions for exact_match, non_embeddable, and embeddable. Instructs the filter to find the entry in cases where extra characters are in front of or behind the entry without spaces.	<ul style="list-style-type: none"> <li>sf@wk3r</li> <li>fawkm</li> <li>fooblahf@Vvk!ngblahfoo</li> </ul>	

**Next Step:** [Testing the Filter](#)

## Testing the Filter

The verification system is used to test the filter either by submitting entries in the Verify Text box that is in the lower left corner of every page under *Filter* or setting up verification cases via *Filter -> Verification*.

When a match is found, it is replaced with asterisks in the result pane. Mousing over the asterisks will display information on the match. Clicking the match will perform a search of the blacklist from the root entry of the match and take you to the search results screen. Click on an entry to access the [add/edit entry](#) screen.



## Troubleshooting False-Positives

When the filter finds a match that it should not have, follow the steps below:

1. Enter the entire message that generated the match in the Verify Text box and click *Submit*.
2. Check for filter matches in the Result column as displayed above. If no matches are present, skip to Other Possibilities below.
3. Mouse over the match to find the details of the match, particularly **Text**. Click the asterisks to perform a search of the blacklist for the entry.
4. Find the entry in results page and click the text to enter the [Add/Edit Blacklist Entry](#) screen.
5. If the reason for the match result has not already been discovered:
  - a. Take a note of the filter mode and review the [Filter Mode](#) descriptions.
  - b. Inspect the values within the Variations box as well as the inflections generated from part(s) of speech by clicking [Show Combinations](#).
  - c. If the reason for the match is still not found, [contact us](#) for assistance.
6. Make adjustments to the entry by adjusting the filter mode, removing variation(s), or [adding inflections to the whitelist](#).
7. Submit changes and retest using the Verify Text box.
8. If the issue has not been resolved, go back to step 3. Else, use the [approval system](#) to make the change live.
9. Test the change in your application or by using the WebService Test Form described below.
  - a. If the change is not taking effect, review Other Possibilities below.

### Other Possibilities

If the match is not present using the Verify Text box in the management interface but the match exists in the application, there may be hidden characters that are sent from your application to CleanSpeak, such as HTML mark-up that can effect filter results. [Contact us](#) for details.

If you made the change in the Management Interface, verified that the change corrected the issue, and approved the change via *Filter -> Approvals*, but the change is not taking effect in your application or when using the WebService Test Form below, then there is a technical issue. [Contact us](#) for assistance.

### WebService Test Form

The WebService test form is a way to test the filter outside of the management interface. This is simulating your application sending content to CleanSpeak to be filtered rather than having to use your application directly. Follow the steps below to access the form:

1. Ask your IT department or dev team for the URL to the WebService. The address will look something like `http://someURL:8001`
2. Append `/test` to the URL to access the WebService test form: `http://someURL:8001/test`
3. If you see a Login WebService Authentication Key message, go back to the Management Interface and retrieve the key via *System -> Configuration*
4. Enter the message to test in the *Message to filter* text box. Do not select any Tags. Click *Filter*.
5. Results will be displayed under *The result*. Any matches found by the filter will be replaced with asterisks.

\*Improvements to this form will be coming soon!

**Next Step:** [Approving Changes](#)

## Approving Changes

Changes to the lists must be approved by a [filter manager](#) or [administrator](#) prior to going into effect via *Filter -> Approvals*. It is recommended that you [test all changes](#) with the verification system prior to approval, but it is not necessary.

Changes may also be reverted which allows flexibility to experiment with the lists without the fear of breaking the database.

**Next Step:** [Whitelist](#)

## Whitelist

### Ignoring variations when blacklist filtering

If you have a case where you want to ignore a variation that CleanSpeak finds automatically, add the variation to the whitelist via *Filter -> Whitelist -> Add Entry*. Add the variation in the Text field and set the two-digit language code.

### Whitelist filtering

Applying CleanSpeak as a whitelist filter as described in the [Filtering Concepts](#) section is managed similarly to blacklist filtering. First, use the management interface to configure the white list which can be [exported](#) to a format used by the client application to filter against while a user is typing. Then, configure **phrases** that should be filtered or alerted on with the blacklist. [Contact us](#) for assistance with implementing whitelist filtering.

**Next Step:** [Import/Export Lists](#)

## Import/Export Lists

Importing Lists – A properly formatted XML file may be imported via Filter -> Import. The imported entries **must be approved** via *Filter -> Approvals*. If you have blacklist that was used with another filtering application that you would like to import, [contact us](#) for details.

You may also export your list into an XML, SQL, or CSV file via *Filter -> Export*.

**Next Step: Multilingual**

## Multilingual

CleanSpeak can filter any language and character set that is supported by the UTF-8 standard. We provide inflection support and filtering lists for the following languages: English, Spanish, French, German, Italian, Russian, Dutch, and Polish. Additional languages may be requested by [contacting us](#) or adding entries yourself with the details below.

### Inflection & Variations

CleanSpeak automatically looks for inflections of root entries and variations when at least one part of speech is selected and the language is supported (see the list of supported languages above). For example, the table to the right displays the list of inflections CleanSpeak will look for from a root entry of **caminar** with a locale of **es** and **verb** part of speech.

If you do not want CleanSpeak to filter a specific inflection, you can add the inflection to the [whitelist](#). For shorter words or special cases, you may also choose to not select a part of speech and use the variation box to add inflections individually.

When adding entries for a language that CleanSpeak does not provide inflection support, add each inflection in the variation box.

For details on specifying the language and using the variation box, visit the [Add/Edit Blacklist Entries](#) page.

caminar			
<ul style="list-style-type: none"><li>• camina</li><li>• caminaba</li><li>• caminabais</li><li>• caminaban</li><li>• caminabas</li><li>• caminad</li><li>• caminada</li><li>• caminadas</li><li>• caminado</li><li>• caminados</li><li>• caminamos</li><li>• caminan</li><li>• caminando</li><li>• caminar</li><li>• caminara</li></ul>	<ul style="list-style-type: none"><li>• caminarais</li><li>• caminaran</li><li>• caminaras</li><li>• caminare</li><li>• caminareis</li><li>• caminaremos</li><li>• caminaren</li><li>• caminares</li><li>• caminaron</li><li>• caminará</li><li>• caminarán</li><li>• caminarás</li><li>• caminaré</li><li>• caminaréis</li><li>• caminaria</li></ul>	<ul style="list-style-type: none"><li>• caminaríais</li><li>• caminaríamos</li><li>• caminarían</li><li>• caminarías</li><li>• caminas</li><li>• caminase</li><li>• caminaseis</li><li>• caminasen</li><li>• caminases</li><li>• caminaste</li><li>• caminasteis</li><li>• caminastes</li><li>• camine</li><li>• caminemos</li><li>• caminen</li></ul>	<ul style="list-style-type: none"><li>• camines</li><li>• camino</li><li>• caminá</li><li>• caminábamos</li><li>• camináis</li><li>• camináramos</li><li>• camináremos</li><li>• caminás</li><li>• caminásemos</li><li>• caminé</li><li>• caminéis</li><li>• caminés</li><li>• caminó</li></ul>

### Preventing False-Positives

When filtering multiple languages in the same application, word collisions may occur. For example, "mama" is an innocent word in English but can be considered vulgar in Spanish. Follow these steps to limit false-positives and address word collisions:

1. Only filter languages that are relevant to your user base.
2. If you have a global application, specify the language for the specific server/region using the [locale request attribute](#).
3. Check the inflections CleanSpeak automatically looks for and use the whitelist to ignore inflections you do not want filtered (see above).
4. Use duplicate entries as described below.
5. (Detailed Integration) In cases where you want to inspect content that has a specific match, such as "mama":
  - a. Create a list of matches to be inspected.
  - b. Provide the list to your development team to parse the [filter response](#) to determine if what was matched is on your list to be inspected.
  - c. If using the CleanSpeak moderation system, resubmit the content to a specific Component designed to review word collisions, or ask your development team to provide an interface to do so.

### Duplicate Entries

Each blacklist entry is defined in CleanSpeak as the base entry + locale. Therefore, you may have separate entries for "smurf" with locales **en**, **en\_US**, **en\_GB**, **es**, etc. The following are cases where having duplicate entries would be useful with multilingual filtering:

### Severity and Tags

A blacklisted word or phrase may have a different severity level and/or tag (category) in various regions/languages. By adding a duplicate entry in the blacklist with a new locale, you can configure the entry differently according to your filtering requirements as described in the [List Management](#)

overview page.

### **Smaller Filtering List**

You may want to filter a small set of a particular language rather than the entire blacklist to prevent generating false-positives. This can be implemented by using the language + country code combination as described in the [Add/Edit Blacklist Entry](#) page. The following example applies specifically when you host a single CleanSpeak installation to filter multiple regions:

#### **Two Global Deployments**

Your application is deployed in two locations, one in America and the other in Germany. Your American deployment will filter English only. For your German user base, you want to filter German and a short list of English profanities.

Add the short list of English entries and set the locale for each to be **en\_DE** (English language, Germany as the region). When sending content to CleanSpeak to be filtered, [specify the locales](#) to be filtered as **en** for the American deployment and **de & en\_DE** for the German deployment.

**Note:** CleanSpeak will look for inflections based on the language code and part of speech as described in the Inflections section above. By specifying a locale to be **en\_DE** and setting at least one part of speech, CleanSpeak will automatically look for English inflections based on the **en** portion of the locale setting.

## **Moderation**

The CleanSpeak moderation system is designed for moderators to easily access the highest priority content first and take actions quickly. Within CleanSpeak, moderators will:

- Action content in real-time
- Review content history
- Take action on users
- Review user history
- Escalate content to managers
- Review conversations in context

All content sent to the CleanSpeak moderation system is stored and can be accessed with the [Content Search](#) form. Content that is prioritized for moderators to review is organized with queues: Approval Queue, Alert Queue, Review Queue, and Escalations (see the [Moderation Concepts](#) section). Queue content is accessed from the [Moderation Dashboard](#).

#### **Next Step: Setup & Configuration**

- [Setup & Configuration](#)
- [Dashboard](#)
- [Queues](#)
- [Escalations](#)
- [Threaded View](#)
- [Users & Actions](#)
- [Content & User Search](#)

## **Setup & Configuration**

[ [Servers](#) ] [ [System Configuration](#) ] [ [Applications & Components](#) ] [ [Component Settings](#) ] [ [Store Options](#) ] [ [Setting Tags to Generate Alerts](#) ] [ [Example Deployment](#) ] [ [Integration Details](#) ]

### **Servers**

The server settings will be a coordinated effort with you and the team that [installed CleanSpeak](#). The CleanSpeak moderation tools rely on storing content. In order to store content, the [Search Engine](#) must be installed. Also, in order to perform moderation functions that effect your application, a [Notification Server](#) must be present. These functions include: Taking actions on users, approving and rejecting content with the Approval Queue, or edit/deletion of content. Ask your IT staff for the URLs of the Search and Notification servers, then add them to the management interface via *System -> Servers*.

### **System Configuration**

### System Configuration

System Default Timezone \*

America/Denver

Authentication Enabled \*

Authentication Key

**System Default Timezone:** Set to the application time zone.

**Authentication:** If CleanSpeak is deployed in a publicly accessible location, Authentication must be enabled. See the [technical documentation](#) for details.

### Moderation Configuration

Queue Size

30

Check Out Minutes

10

**Queue Size:** When accessing queues from the dashboard, moderators will check out a set amount of content to be displayed on their screen. This setting determines how many items they will see at a time, with a default value of 30.

**Check Out Minutes:** Once content is checked out from a moderator, no other moderator will be able to access the content by clicking the queue from the dashboard. The content will be checked back in if the moderator does not take action on the content within this time frame, with a default value of 10 minutes.

## Applications & Components

At least one Application and one Component must be configured to apply the moderation features. Review the [Moderation Concepts](#) page to determine which applications and components are appropriate for your application. An [example deployment](#) is described below.

### Component Settings

Check *Store Content* to begin. By checking either *Approvals Enabled?* or *User Actions Enabled?*, the Notification Server setting will become visible. The Notification Server must be active in order for CleanSpeak to send information to your application(s). See the [Servers](#) section at the top of this page for details.

#### Store Options

- **Store only:** Use this option if you do not want to activate the Alerts or Pending Review queues.
- **Store and review all content:** Use this option to activate the Pending Review queue.
- **Store and review only content that generates alerts:** Use this option to only review categories of content with the Alert Queue.

#### Setting Tags to Generate Alerts

Alerts are generated when the filter finds a match on a blacklist entry with specific tags. Configure which tags you would like to generate alerts from for each component.

Approvals Enabled?

User Actions Enabled?

#### Notification Servers

http://localhost:8011/mock-notification

store only

store and review all content

store and review only content that generates alerts

#### Alert on tags matching:

AsciiArt

Bullying

Competitors

Drug & Alcohol

Gang

Grooming

Profanity

Racial

Religious

Weapons

Youth

### Example Deployment

In the following example, CleanSpeak is deployed to moderate content for two games:

	AdultGame	ChildrensGame
<b>Description</b>	Targeted and rated for mature audiences only	Targeted for kids and tweens.
<b>User-Generated Content Sources</b>	In-game chat, forum posts, username registration	In-game chat, forum posts
<b>Moderation Concepts</b>	<ul style="list-style-type: none"><li>• Chat: Rely on the filter, no active moderation</li><li>• Forum Posts: Be warned when submissions contain racist or religious remarks</li><li>• Usernames: Review all usernames that have been submitted</li></ul>	<ul style="list-style-type: none"><li>• Chat: Review every chat message that is submitted</li><li>• Forum Posts: Review each post with an approval process prior to being displayed publicly</li></ul>

The following list includes the settings for each Application and Component as described above:

- Application Name: **AdultGame**
  - Component Name: **Chat**
    - Settings: Store only
  - Component Name: **Forum**
    - Settings: Store and review only content that generates alerts. Tags selected: Racial, Religious
  - Component Name: **Usernames**
    - Settings: Store and review all content
- Application Name: **ChildrensGame**
  - Component Name: **Chat**
    - Settings: Store and review all content
  - Component Name: **Forum**
    - Settings: Approvals Enabled (checked), Store only

### Integration Details

In the previous example, once the applications and components are configured, you will send the following information to the development team:

- Each application and component name.
- Instructions that content from *ChildrensGame: Forum* will require approval.

The development team will implement the [moderation API](#) to deliver content from each source correctly.

**Next Step:** [Moderation Dashboard](#)

## Dashboard

The moderation dashboard displays each Application and Component that is available to the moderator that is logged in. Escalations are also displayed if logged in as a moderation manager or administrator.

Total content stored for each component is shown at the top of the component window. If there are pending escalations, alerts, reviews, or approvals, there is a link to access the respective queue.

Clicking a queue count will check out content for the specific moderator that is logged in. If a different moderator is logged in and clicks the same queue, a separate set of content will be checked out. If the content is not actioned in time, it will be automatically checked back in for another moderator to access. The content count and check out duration is set in the [system configuration](#).

**Next Step:** [Queues](#)

## Queues

Content in queues requires an action to be taken. Pending Alert & Pending Review queues requires that the content is acknowledged by clicking Dismiss All at the bottom of the page. Pending Approvals requires the moderator to Approve, Reject or Ignore the content. From any type of queue, the moderator may also [Escalate](#) content to a manager.

Each content item will display a link to the user that generated the content, a [Threaded View](#) link, date/time when the content was generated, the Location, and the message. Clicking the user will open a new tab or window depending on your browser settings so that the user details and history can be reviewed without leaving your queue. Threaded View will pop up a window displaying the conversation before and after the content as determined by the Location (see [Threaded View](#)).

## Content Display Order

Content is sorted in each queue by the severity level of matches found by the filter (if any) so that moderators are always reviewing the higher priority content items first. Content that contains matches from the blacklist will be at the top of the queue to be checked out prior to content that has no matches. For content that has an equal severity level, the display order is sorted chronologically, where the oldest content is checked out first.

**Next Step:** [Escalations](#)

## Escalations

Content may be escalated to a moderation manager by a moderator in cases where the moderator is not sure what action should be taken.

Escalations may be submitted from any queue. A comment may be added to an escalation but is not required. Once submitted, content that is escalated will immediately be removed from the current queue and reside in the escalations queue for a manager. The escalated content is still viewable by the moderator if needed via *Moderation -> Content Search*.

Escalate

Comment

This user has mentioned eating carrots many times in the last hour. Is there something else going on?

Save

Cancel

**Next Step:** [Threaded View](#)

## Threaded View

When content is [sent to CleanSpeak for moderation](#), the Location is optionally set for Threaded View. Location could be a room in a virtual world, a forum thread, a chat channel, a private conversation, etc. Displaying a conversation in context with Threaded View is available from any piece of content displayed via queues or content search that has a Location associated with it. A separate window will pop up when clicking Threaded View that displays content from all users within the same Location before and after an adjustable time range.

### Overview

Within the Threaded View window, the original message will be outlined with a dark black border. All messages sent from the Original Sender (the sender that is associated to the piece of content that *Threaded View* was clicked from) will have a light gray background. All other messages will have a white background. As you scroll through the messages, you can click *Original Message* link at anytime to regain focus on the original message.

Clicking on a user will open a new tab or window depending on your browser settings so that the user details and history can be reviewed without leaving your queue.

To save a record of the conversation, you can copy/paste the conversation by highlighting it with your mouse, copying, and pasting to any word processor.

### Implementation

First, determine what conversations you will want to see in context. As mentioned above, this could be one-to-one messages or room/channel conversations in real-time chat applications. In non-chat applications, the context might be individual form threads, blog commenting threads, etc.

Next, hand off a list of these contexts which will then be used as Locations for the development team. Each context/location should be accompanied with an [Application and Component](#) for clarity. The development team will use this information to send content to CleanSpeak correctly with the [moderation API](#).

Time Range:   [Original Message](#)  
 Message from Original Sender  
 Message from Other

[SketchyGuy](#) (02/13/2012 09:50:26 AM MST) "hey there"

[YoungPerson](#) (02/13/2012 09:50:39 AM MST) "hey"

[SketchyGuy](#) (02/13/2012 09:50:53 AM MST) "good to see you online again"

[SketchyGuy](#) (02/13/2012 09:51:00 AM MST) "how was your day?"

[YoungPerson](#) (02/13/2012 09:51:08 AM MST) "it was ok"

[SketchyGuy](#) (02/13/2012 09:51:23 AM MST) "just ok?"

[YoungPerson](#) (02/13/2012 09:51:34 AM MST) "yeah, could be better i guess"

[SketchyGuy](#) (02/13/2012 09:51:57 AM MST) "I have an idea. r your parents home?"

[YoungPerson](#) (02/13/2012 09:52:09 AM MST) "yes, but they're leaving soon"

[SketchyGuy](#) (02/13/2012 09:54:14 AM MST) "can I see u?"

[YoungPerson](#) (02/13/2012 09:54:29 AM MST) "um, really?"

**Next Step:** [Users & Actions](#)

## Users & Actions

User details and history are available by clicking a user from a content display or result of a User Search. Profile information includes all user data that is sent to CleanSpeak. From the user detail screen, moderators can add comments about the user, review the user's latest content, review all content that contained a match from the filter, take actions on the user, and review the event history of the user.

### Moderation – User Actions

Moderators can take actions on users directly from CleanSpeak such as sending the user a warning message or kicking them from the application for a period of time. The actions are customizable and each has a key associated with it.

Keys are typically going to be time values or message names. In this screenshot, the action is *Warn* with keys consisting of pre-formatted messages that can be sent quickly. The messages themselves are written and stored with the client application. Actions such as Kick and Silence will have time values as keys (15 minutes, 30 minutes, 2 days, etc).

User actions and keys are configured via *System -> User Actions*. Once defined, coordination with the IT staff is required for integration with your application.



**Take an Action**
HIDE

Action

Key

Components

Comment

## Content & User Search

Comprehensive content and user search forms are available via *Moderation -> Content Search* and *Moderation -> User Search*. Results from the content search form will include the moderation status and moderation action when applicable.

## FAQ & Troubleshooting

### *How do I know what filter mode to use?*

- Details and examples can be reviewed in the [Filter Mode](#) section of the [Add/Edit Blacklist Entry](#) page.
- **Distinguishable Best Practices:** Avoid false positives when using distinguishable by performing a whitelist search of the base entry. For example, a whitelist search of “anal” generates 153 results of words that contain “anal”, such as analeptic and canal. Entries on the whitelist will be ignored as long as they are spelled correctly. However, If “anal” is distinguishable and the user types “analeptic” (misspelled), the filter will generate a match that is considered a false positive. Therefore, “anal” should be set to embeddable instead. An entry like “fawk”, on the other hand, could be set to distinguishable without the danger of creating false positives since there are zero dictionary words that contain the string “fawk”.

### *Why is something being filtered in my application that does not get filtered with the verification tool?*

- First, check to see if there are pending approvals via *Filter -> Approvals*. Any change to the lists that is pending approval will not be filtered in the application. Also, be sure there are no hidden characters being sent to CleanSpeak by the application. Rich text editors may include html tags, for example, that will effect the filter. More information on diagnosing false-positives is available in the [Testing the Filter](#) section.

### *Why is the word I added not taking effect in my application?*

- The change has not yet been approved via *Filter -> Approvals*. If your role does not give you the ability to approve list changes, have a filter manager or moderator [approve the change](#). If the change is approved and still not taking effect, there may be a technical issue. [Contact us](#) for assistance.