Documentation

Search All 2.3 Documentation

Welcome to the CleanSpeak 2.3 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 API section of the technical documentation to send content to CleanSpeak with the configuration provided by the Project Manager from the previous step.

For AWS users, an AMI is also available for the Installation step. Please visit the AWS AMI page for details.

Default Filtering Lists

The default black list that is initially installed contains many entries that are specific to filtering children's applications. Review the List Management section for a description of each severity level and tag set.

If your application is targeted to a mature audience, you may want to remove entries from the black list that are specific to younger audiences. Visit the Blacklist Search page for details.

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.

What is the default login information for the Management Interface?

- username: **admin@inversoft.com**
- password: **password**

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 Examples section. If you'd like detailed integration services, please contact us at support@inversoft.com.

Release Notes

2.3.21

Released Wednesday December 2, 2015

- Back-ported fix from 3.1.3. Enhanced our ability to find repeating replacement characters when separated by punctuation.

2.3.20

Released Tuesday December 1, 2015

- Back-ported fix from 3.1.3. Enhanced our ability to find repeating characters when separated by punctuation.

2.3.19

Released Wednesday September 2, 2015

This version requires a database migration

- Fixed possible error in filter approvals that can cause blacklist entries to lose all their tags
- Added timestamp to audit logs
- Fixed various date and time displays to use the selected timezone

2.3.12

Released Friday July 18, 2014

- Updated filter replacement characters to cover more unicode characters.
- Updated filter to be more aggressive finding blacklist entries that contain numbers when the entry is marked as Distinguishable.

2.3.11

Released Friday July 11, 2014

- Added intelligent filtering support for Danish, Norwegian, and Finnish languages.

2.3.9 & 2.3.10

Released Thursday July 3, 2014

- Updated the replacement string value in a moderation response to reflect configuration set in Filter Rules.

2.3.8

Released Friday June 20, 2014

- Repaired an issue where embedded punctuation would prevent phrases from being filtered properly.
- Repaired an issue where the filter wasn't correctly returning all results from multilingual matches that contain special characters.

2.3.7

Released Thursday March 13, 2014

- Increased filter accuracy in cases of overlapping blacklist matches.

2.3.6

Released Monday March 3, 2014

- Updated handling of SAML exceptions.

2.3.5

Released Thursday February 20, 2014

- Repaired an issue where content that is edited from a moderation queue could not be deleted.

- Updated blacklist filter accuracy.

2.3.4

Released Friday February 14, 2014

- Updated handling of SAML exceptions.

2.3.3

Released Sunday February 9, 2014

- Repaired a bug with XML signatures for the Single Sign On feature.
- Updated the Escalations feature in the management interface to handle complex content properly.

2.3.2

Released Thursday November 21, 2013

- Repaired a bug that caused Reports not to be generated properly.

2.3.1

Released Tuesday October 29, 2013

- Repaired an error with the audit log where list additions were not being logged properly.
- Filter -> Import now works properly with client browsers running in Windows.

2.3

Released Friday October 18, 2013

A database migration is required and available in the downloads section of your account: http://www.inversoft.com/login.

- The configuration and license file system for CleanSpeak has been completely rewritten to make your life simpler. Check out the Configuration and License pages for details.
- Added whitelist (restrictive) chat filtering.
- Added support for complex content. Content can now consist of multiple pieces such as text, images and video. See the Content Item API for details.
- Added configuration in the Management Interface for moderation managers to control the filter and moderation work flows. See: Moderation Setup & Configuration and updates to the API Response Reference (search for contentAction).
- The blacklist filter has been updated to improve accuracy and control.
- We transitioned from Tomcat's version of DBCP to BoneCP for our database connection pool. BoneCP will be a much more reliable and resilient connection pool.
- Added SSL support.
- Too many bug fixes to list here, but many blacklist filter bugs have been fixed and many interface bugs have been fixed.

Technical Guide

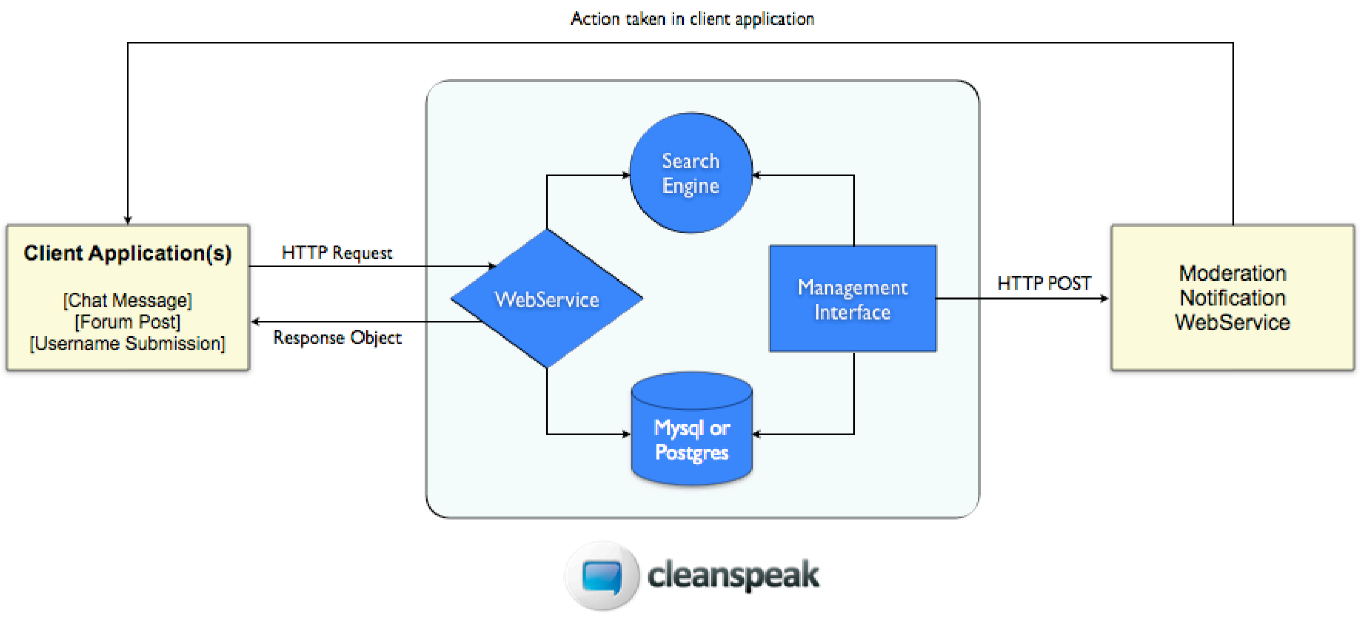
Welcome to the CleanSpeak 2.3 Technical documentation.

Search this guide:

Getting Started

CleanSpeak Overview

CleanSpeak is an on-premise software solution to filter and moderate user-generated content. Developed in Java, the CleanSpeak system is composed of four components: WebService, Management Interface, Search Engine, and a Database. Each component may be installed on a single server/VM. The system scales by load balancing WebServices for throughput and Search Engines for indexing content. Throughput, requirements, and other metrics are described below.



The **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. These functions include: Filtering text, storing/queueing content for moderator review, and storing user details.

The **Management Interface** is a graphical user interface to manage filtering lists, review and search for content, review user details, and perform actions that have an immediate effect in the client application, such as deleting forum posts or kicking users.

The **Search Engine** indexes content from the database for immediate, full-text search abilities within the Management Interface.

The **Database** is a relational database. We currently support mysql and postgres.

The **Moderation Notification Webservice** is developed by the client to receive action notifications from the Management Interface and take the appropriate action in the client application.

Requirements & Metrics

Operating System	CleanSpeak will run on most modern operating systems that are capable of running Java version 1.6 or higher. Install and update packages are available in RPM, DEB, and ZIP versions.

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 Oracle • OpenJDK 6.0 or higher • JDK 6.0 or higher from Apple
Database	Mysql version 5.1 and above or PostgreSQL version 9.1 and above.
Throughput	Varies with hardware and network systems, ranging between 10,000 -> 20,000 messages per second (1 billion -> 2 billion per day). Scales horizontally as needed.

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 guide. 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 7.0.40

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.

- MySQL 5.0 or higher
- PostgreSQL 9.1 or higher

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 configure CleanSpeak (which includes logging into your account at Inversoft Account and copying your License ID into the CleanSpeak configuration). Configuring CleanSpeak is covered here:

- Configuration

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 named **JAVA_HOME** (if you installed the JDK) or **JRE_HOME** (if you installed the JRE) 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 msvcr71.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 installed 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 'cleanspeak'"
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 -pcleanspeak 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 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. This file contains data to filter English only. Once you have downloaded this file and extracted it, execute this command:

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

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

Additional Languages

The real-time filter data described in the previous step installs filtering data for English only. Data for filtering in additional languages is available by downloading the cleanspeak-database-languages ZIP file. This contains JSON files of additional languages that can be imported from the management interface once it is installed and running. See: Import/Export Lists.

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 7.0.40
- 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 7.0.40
- 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 7.0.40
- (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 CLEANSPEAK_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-7.0.40\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 7.0.40, 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 7.0.40, 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:

- 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 7.0.40
- 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 7.0.40
- 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 7.0.40
- (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-7.0.40\bin>serv
ice.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 7.0.40, 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-management-interface-<version>.deb
```

This installation process places the CleanSpeak Management Interface, including Tomcat 7.0.40, 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:

- 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 7.0.40
- 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 7.0.40
- 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 7.0.40
- (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 CLEANSPEAK_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-7.0.40\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 7.0.40, in the directory /usr/local/inversoft/cleanspeak-search-engine. 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-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 7.0.40, 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.

Index Directory

By default, the solr index home directory (where the index is stored) defaults to the apache tomcat 'bin' directory (where the executable lives). We recommend you change this to a mounted filesystem with daily backups. You can configure this directory in the cleanspeak.properties file as follows:

LINUX:

> *cleanspeak-search-engine.data-directory=/usr/local/inversoft/cleanspeak-search-engine/data*

WINDOWS:

> *cleanspeak-search-engine.data-directory=C:\\inversoft\\cleanspeak-search-engine\\data*

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:

- 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/cleanspeak-management-interface-{VERSION}/apache-tomcat-7.0.40/lib and CLEANSPEAK_HOME/cleanspeak-webservice-{VERSION}/apache-tomcat-7.0.40/lib directories.

License ID

Configuration

You need to configure your License ID in the cleanspeak.properties configuration file. You can find your License ID by logging in to http://www.inversoft.com/login.

CleanSpeak will periodically perform a call-back to Inversoft to retrieve your license details. This call-back is performed over port 80.

Optional License File Manual Installation

This feature is only available for clients that have licensed CleanSpeak annually. However, we recommend that all customers use our network based licensing to make renewals and license management simpler.

If you would prefer prevent CleanSpeak from calling back to the Inversoft servers, you can manual your license file manually. As long as you have purchased a yearly of CleanSpeak license and your account is up-to-date, contact Inversoft support to enable license file downloads on your account. After license file downloads have been enabled, you can login to your account at http://www.inversoft.com/login to download your license file. Click the Download CleanSpeak 2.3+ License File from your account page and install your license file into the /config directory as specified below:

Linux

```
/usr/local/inversoft/config/<Your License ID>.license
```

Windows

```
%CATALINA_HOME%\..\..\config    -> If you install to C:\inversoft this
directory will be C:\inversoft\config\<Your License ID>.license
```

You will still need to specify your License ID in the cleanspeak.properties file in order for CleanSpeak to locate the correct license file and load it. You can read more about this configuration on the Configuration page.

Updating

When manually managing your license file, you will need to replace your license file when you renew each year (or if something else changes on your account). Download an updated license file and install as described above. The CleanSpeak services do not need to be restarted for the update to take effect.

CleanSpeak will automatically update your license file when implementing the call-back method described at the top of this page when your account is updated. You do not need to do anything to your CleanSpeak installation for account changes (renewals, etc) to take effect.

AWS AMI

Overview

A public AWS AMI is available that has the latest version of CleanSpeak 2.3 Filtering & Moderation Tools installed. The image consists of:

- Ubuntu 14.04 LTS
- Java OpenJDK 1.7.0_51
- PostgreSQL and MySQL clients for database management from the command line
- Each CleanSpeak service installed via dpkg: Management Interface, Webservice, and Search Engine

The image does not contain a database. You can install a database on the image after creating it or use a database from a separate instance (RDS recommended). Review the Database Installation documentation for details.

An EBS volume is required due to the CleanSpeak Search Engine that requires persistent storage. For filter-only applications of CleanSpeak, instance only storage may be used. Inversoft does not currently provide an AMI for filter-only applications with instance only storage, but this AMI (with EBS storage) can be used. The CleanSpeak Search Engine package can either be removed or the service simply not started.

Installation

The following instructions are for setting up a single instance. To run multiple instances, see the Scaling and Redundancy documentation.

Launch a New Instance

Log into your AWS Management Console and navigate to the EC2 Management Console. Create a new EC2 instance from the CleanSpeak AMI with one of the following methods:

- Click **Instances** then **Launch Instance**. Select **Community AMIs** and search for *CleanSpeak*. Click **Select** for the latest CleanSpeak 2.3 AMI displayed.
 OR
- Click **AMIs** and change the Filter ***Owned by me*** to ***Public Images***. Search for *CleanSpeak*, select the latest 2.3 instance, and click **Launch**.

The CleanSpeak AMI is public for each US region. If you're looking for the AMI in a region outside of the US, contact support@inversoft.com.

Details to launch an AWS EC2 Instance can be found from the AWS documentation and will not be repeated here. Relevant notes:

- When running each CleanSpeak service in a single instance, an m1.small fulfills minimum requirements for development and evaluation. However, an m3.medium instance (or larger) is recommended for production, staging, etc. See the Requirements documentation for details.
- The default size of the EBS volume for the AMI is 20GB, which can be increased if a large search index is anticipated or if other applications will be installed on the instance.

Configure CleanSpeak

Connect to the CleanSpeak instance by navigating to **Instances** from the EC2 Management Console and click **Connect**.

Add your license ID and database credentials to the cleanspeak.properties file located in /usr/local/inversoft/config as described in the Configuration section. The database must be installed prior to this step.

The default memory allocated for each service is 512M to accommodate installations on m1.small instances. For medium and greater instances, we recommend changing the value for each CleanSpeak service to 1G or higher.

Restart the CleanSpeak Services

When the instance is first created and launched, each CleanSpeak service will be started but CleanSpeak will not be running properly yet because the configuration wasn't set as described above. First, stop each service:

```
sudo service cleanspeak-management-interface stop
sudo service cleanspeak-webservice stop
sudo service cleanspeak-search-engine stop
```

Then, clear out the CleanSpeak log files:

```
sudo rm /usr/local/inversoft/logs/*.log
```

Finally, restart each service:

```
sudo service cleanspeak-management-interface start
sudo service cleanspeak-webservice start
sudo service cleanspeak-search-engine start
```

Note:

- For filter-only applications, do not start the cleanspeak-search-engine service. The service can be completely removed if desired by the following command:

  ```
  sudo dpkg -P cleanspeak-search-engine
  ```

Test that CleanSpeak is running properly

The following curl command will send a simple request to the webservice and is followed by the expected response:

***Note**: If you've included an authentication token in the cleanspeak.properties file, be sure to add **-H "Authentication: <token>"** to the curl request.

```
% curl -d
"contentItem.content=fuck&contentItem.type=text&filter.blacklist.enabled=t
rue" http://localhost:8001/content/item.js
{
  "filter": {
      "matched": true,
      "matches": [
            { "start": 0,
              "length": 4,
              "type": "blacklist",
              "tags": ["Vulgarity"],
              "severity": "severe",
              "locale": "en",
              "matched": "fuck",
              "root": "fuck",
              "quality": 1.00000 }
      ],
        "replacement": "****"
  }

}
```

The Management Interface and Search Engine may be tested with the following curl commands that will generate an html response:

```
// Management Interface
curl -v "http://localhost:8011/login"

// Search Engine
curl -v "http://localhost:8021/#/"
```

If CleanSpeak is not running properly, visit the Troubleshooting section of our documentation.

Configure Your Security Group

In order for CleanSpeak to handle incoming requests, the instance security group must be configured to allow incoming TCP connections from port 8011 for the Management Interface and port 8001 for the Webservice (you may chose to open all TCP incoming ports or just 8001 and 8011). If you allow incoming requests from the public (any IP address), you **MUST** enable CleanSpeak HTTP Authentication.

See the AWS Security Group Documentation for details.

Integrate with CleanSpeak

Installation is complete! Now, begin integrating with CleanSpeak by setting up your moderation system (for moderation clients) or hitting the Conte nt API immediately for filter-only applications.

Useful notes and links:

- The default admin credentials to login to Management Interface:
  - admin@inversoft.com
  - password
- Managing the Filter Lists
- Integration Examples

Updates & Patches

When updates become available, you will need to perform the updates directly (always recommended but never required). The process is very quick and is described in the Linux (Debian) section of Updates & Patches. You will be informed when updates are available via email as well as the post-login screen of the Management Interface.

Other JEE Web Servers

Installation – Other Web Server

All of the bundles (except the webapp only bundle) come with Tomcat 7.0.40. This is the preferred web server. If you need to run CleanSpeak 2.3 in a different JEE WebServer, you need to contact Inversoft to assist with your specific needs.

Configuration

Configuration

CleanSpeak is configured entirely in a single file. The name of this file is **cleanspeak.properties** and it lives in the directory /usr/local/inversoft/config on Linux or C:\inversoft\config on Windows (if you installed CleanSpeak to C:\inversoft). This file is preconfigured to run without modification (with the exception of the License ID) with a standard installation. However, you can modify this file with any of the configuration properties listed below.

This file is a standard Java properties file that uses key value pairs like this:

```
database.url=jdbc:mysql://localhost:3306/cleanspeak
```

Values can span multiple lines using a backslash character like this:

```
key=value,\
  another value,\
  last value
```

For more information on the properties file format, visit this Wikipedia entry: http://en.wikipedia.org/wiki/.properties

Required Properties

Property Name	Description
license.id	Your License ID. You can find your License ID by logging into your account at http://www.inversoft.com/login. This property is required regardless of the license method you are using. For most customers, CleanSpeak will use this License ID to download your license information from Inversoft's license server. You don't need to install any license files or manage renewals or updates by hand. CleanSpeak will manage all of this for you and when your account is renewed and automatically download your new license details. If you are prevented from using our network licensing system, you can manually install a license file by following the directions at the bottom of this page.
database.url	A JDBC URL for your database. For MySQL, this URL will look something like this: jdbc:mysql://localhost:3306/cleanspeak For PostgreSQL, this URL will look something like this: jdbc:postgresql://localhost:5432/cleanspeak
database.username	The username used to connect to the database.
database.password	The password used to connect to the database.

email.host	The hostname of your email server. This property is used by CleanSpeak for allowing moderators to reset their passwords via the Management Interface.
email.port	The post for your email server (standard is 25).
email.ssl	Whether or not CleanSpeak should use SSL to connect to your email server.
email.username	The username used to connect to your email server.
email.password	The password used to connect to your email server.
email.from	The email address used as the From: when sending new emails. Make sure that your email server will accept email with this email address as the sender.
cleanspeak-search-engine.urls	A comma-separated list of URLs for your CleanSpeak Search Engine instances. These should be properly formatted HTTP URLs like this: cleanspeak-search-engine.urls=http://server1:8021/,http://server2:8021/
cleanspeak-webservice.urls	A comma-separated list of URLs for your CleanSpeak WebService instances. These should be properly formatted HTTP URLs like this: cleanspeak-search-engine.urls=http://server1:8011/,http://server2:8011/
cleanspeak-management-interface.http-port	The port number that the CleanSpeak Management Interface will use for its main HTTP requests (via the a web browser).
cleanspeak-management-interface.https-port	The port number that the CleanSpeak Management Interface will use for its main HTTPS requests (via the a web browser).
cleanspeak-management-interface.management-port	The port number that the CleanSpeak Management Interface will use for Tomcat's internal administration.
cleanspeak-search-engine.http-port	The port number that the CleanSpeak Search Engine will accept requests from the other CleanSpeak services on.
cleanspeak-search-engine.https-port	The port number that the CleanSpeak Search Engine will accept requests from the other CleanSpeak services on (via SSL).
cleanspeak-search-engine.management-port	The port number that the CleanSpeak Search Engine will use for Tomcat's internal administration.
cleanspeak-webservice.http-port	The port number that the CleanSpeak WebService will use for its main HTTP requests.
cleanspeak-webservice.https-port	The port number that the CleanSpeak WebService will use for its main HTTPS requests.
cleanspeak-webservice.management-port	The port number that the CleanSpeak WebService will use for Tomcat's internal administration.
authentication.system	The authentication system to use. The default value for this property is ***database***. You can set this to ***samlv2*** to enable SAML SSO integration. Consult the Single Sign On page for more information on configuring SAML SSO.

Optional Properties

Depending on your use of CleanSpeak and deployment needs, you might also need to change these configuration properties as well:

Property Name	Description

notification.servers	A comma-separated list of valid JSON objects that describe your notification servers. The JSON objects must have at least a **url** property. They can also optionally have properties for controlling the security of your notification server. Here is an example:
	
	notification.servers= \
	{"url": "http://localhost:7654/notify", "httpUsername": "foo", "httpPassword": "pass"}, \
	{"url": "http://localhost:6543/notify", "awsAccessKey": "key", "awsSecretAccessKey": "secret"}

authentication-token	The API authentication token that your CleanSpeak WebService will use to authenticate requests. This is required if you are running CleanSpeak on a publicly accessible WebService. You can also use this property if you want to secure your WebService for any reason.
cleanspeak-search-engine.data-directory	The fully qualified path to the directory that CleanSpeak will store the Search Engine's data files in.
cleanspeak-webservice.persistent-queue-directory	The fully qualified path to the directory that CleanSpeak should store in its persistent queue in.
cleanspeak-management-interface.memory	The amount of memory to dedicate to the CleanSpeak Management Interface (M for megabytes and G for gigabytes). • For example, 1G will assign 1 gigabytes. • You can't input fractional values such as 1.5G. Instead, use 1500M
cleanspeak-search-engine.memory	The amount of memory to dedicate to the CleanSpeak Search Engine (M for megabytes and G for gigabytes). • For example, 1G will assign 1 gigabytes. • You can't input fractional values such as 1.5G. Instead, use 1500M
cleanspeak-webservice.memory	The amount of memory to dedicate to the CleanSpeak WebService (M for megabytes and G for gigabytes). • For example, 1G will assign 1 gigabytes. • You can't input fractional values such as 1.5G. Instead, use 1500M
cleanspeak-management-interface.additional-java-args	Any additional arguments that you want to pass to the Java Virtual Machine that runs the CleanSpeak Management Interface.
cleanspeak-search-engine.additional-java-args	Any additional arguments that you want to pass to the Java Virtual Machine that runs the CleanSpeak Management Interface.
cleanspeak-webservice.additional-java-args	Any additional arguments that you want to pass to the Java Virtual Machine that runs the CleanSpeak Management Interface.

Optional License File Manual Installation

This feature is only available for clients that have licensed CleanSpeak annually. However, we recommend that all customers use our network based licensing to make renewals and license management simpler.

If you would prefer prevent CleanSpeak from calling back to the Inversoft servers, you can manual your license file manually. As long as you have purchased a yearly of CleanSpeak license and your account is up-to-date, contact Inversoft support to enable license file downloads on your account. After license file downloads have been enabled, you can login to your account at http://www.inversoft.com/login to download your license file. Click the Download CleanSpeak 2.3+ License File from your account page and install your license file into the /config directory as specified below:

Linux

```
/usr/local/inversoft/config/<Your License ID>.license
```

Windows

```
%CATALINA_HOME%\..\..\config    -> If you install to C:\inversoft this
directory will be C:\inversoft\config\<Your License ID>.license
```

You will still need to specify your License ID in the cleanspeak.properties file in order for CleanSpeak to locate the correct license file and load it. You can read more about this configuration on the Configuration page.

NOTE: You will need to manually update your license file when you renew your account or if your account details change.

SSL (HTTPS)

By default, CleanSpeak uses a self-signed certificate generated by Inversoft and responds to HTTPS requests on port 8003 for the WebService and 8013 for the Management Interface. In order to prevent browser warnings about the certificate, you will need to purchase a signed SSL certificate from a certificate authority (CA).

NOTE: The keystore password is ***changeit***

Here are the steps you need to follow to get CleanSpeak setup with a valid certificate:

1. Delete the self-signed certificate in the keystore that ships with CleanSpeak using this command:
   a. Windows users (change the location of the keystore depending on where you installed CleanSpeak):

      ```
      C:\> keytool -delete -alias tomcat -keystore
      C:\inversoft\config\keystore
      ```

   b. Unix users:

      ```
      $ keytool -delete -alias tomcat -keystore
      /usr/local/inversoft/config/keystore
      ```

2. Create a new certificate in the keystore. This will replace the self signed certificate that CleanSpeak ships with. Depending on your network setup and the Ceritificate Authority you will be using, you might need to enter the fully qualified domain name to the CleanSpeak server as the ***first and last name*** part of this certificate information. Here is the command to create a new certificate:
   a. Windows users (change the location of the keystore depending on where you installed CleanSpeak):

      ```
      C:\> keytool -genkey -alias tomcat -keyalg RSA -keystore
      C:\inversoft\config\keystore
      ```

   b. Unix users:

      ```
      $ keytool -genkey -alias tomcat -keyalg RSA -keystore
      /usr/local/inversoft/config/keystore
      ```

3. Create a Certificate Signing Request (CSR) and output it to a file named certreq.csr using these steps:
   a. Windows users (change the location of the keystore depending on where you installed CleanSpeak):

      ```
      C:\> keytool -certreq -keyalg RSA -alias tomcat -file certreq.csr
      -keystore C:\inversoft\config\keystore
      ```

   b. Unix users:

      ```
      $ keytool -certreq -keyalg RSA -alias tomcat -file certreq.csr
      -keystore /usr/local/inversoft/config/keystore
      ```

4. Use this CSR to request the certificate from the Certificate Authority (CA). You should receive your certificate and a chain certificate from the CA. You now need to install each one into the keystore.
5. Install the chain certificate into the keystore using this command:
   a. Windows users (change the location of the keystore depending on where you installed CleanSpeak):

```
C:\> keytool -import -alias root -keystore
C:\inversoft\config\keystore -trustcacerts -file
<filename_of_the_chain_certificate>
```

b. Unix users:

```
$ keytool -import -alias root -keystore
/usr/local/inversoft/config/keystore -trustcacerts -file
<filename_of_the_chain_certificate>
```

6. Install your certificate into the keystore using this command:
   a. Windows users (change the location of the keystore depending on where you installed CleanSpeak):

```
C:\> keytool -import -alias tomcat -keystore
C:\inversoft\config\keystore -file <your_certificate_filename>
```

b. Unix users:

```
$ keytool -import -alias tomcat -keystore
/usr/local/inversoft/config -file <your_certificate_filename>
```

After you have successfully imported the certificates into the keystore, restart CleanSpeak and it will use your new certificate.

Single Sign On

SAML v2.0

CleanSpeak currently supports SAML v2.0 for Single Sign On (SSO). In order to enable SAML v2.0 SSO, you need to set the configuration property ***authentication.system*** as follows:

```
authentication.system=samlv2
```

This will enable SAML v2.0 login and role management within CleanSpeak. There are a number of other configuration properties you need to set. The table below lists these properties.

Property Name	Description
authentication.samlv2.login-url	This is the URL of your SAML v2.0 IDP login server.
authentication.samlv2.response-verification-certificate-file	The absolute path to a certificate file that CleanSpeak will use to verify any XML signa response. If this property is not set, CleanSpeak will not verify signatures.
authentication.samlv2.single-attribute-for-roles	The name of an attribute in the SAML v2.0 response that controls what CleanSpeak r Often, SAML v2.0 IDPs will return a multi-value attribute that contains the LDAP grou This property indicates which attribute contains those LDAP groups (or any other type

authentication.samlv2.single-attribute-for-roles-mappings	A JSON object that maps attribute values from the SAML v2.0 response to CleanSpe Often, SAML v2.0 IDPs will return a multi-valued attribute that contains the LDAP gro Those LDAP groups need to be mapped to CleanSpeak roles. This mapping is contro
	
	The JSON object looks something like this:
	
	`authentication.samlv2.single-attribute-for-ro`
	`\`
	` "CN=admin-role,OU=SomeGroup,DC=your,DC=idp,D`
	`["admin"], \`
	` "CN=moderator-role,OU=SomeGroup,DC=your,DC=i`
	`["moderator"] \`
	`}`

Example Configuration

Here is an example configuration for SAML v2.0

```
authentication.system=samlv2
authentication.samlv2.login-url=https://you.idp.com/ssoLogin
authentication.samlv2.response-verification-certificate-file=/usr/local/in
versoft/config/certificate.txt
authentication.samlv2.single-attribute-for-roles=MEMBEROF
authentication.samlv2.single-attribute-for-roles-mappings={\
  "CN=admin-role,OU=SomeGroup,DC=your,DC=idp,DC=com": ["admin"],\
  "CN=moderator-role,OU=SomeGroup,DC=your,DC=idp,DC=com": ["moderator"] \
}
```

POST Back URL

CleanSpeak uses the URI ***/samlv2*** to handle the SAML v2.0 response POST back. CleanSpeak also ships with a self-signed certificate and is configured to respond to HTTPS requests on port 8013. You will need to configure your IDP with this URL as the response post-back URL. If you have setup a DNS name for CleanSpeak, this URL might look something like:

```
https://cleanspeak.your-company.com:8013/samlv2
```

If you wish to use a certificate issued from a certificate authority for CleanSpeak, follow the instructions in the SSL (HTTPS) documentation page to install a signed certificate that CleanSpeak will use for SSL.

Running

Running CleanSpeak

Below are instructions for starting and stopping the CleanSpeak Management Interface, WebService, and Search Engine on both Windows and Linux. For the examples, substitute the following for each respective service denoted by <cleanspeak-service>:

- cleanspeak-management-interface
- cleanspeak-webservice
- cleanspeak-search-engine

Windows

To start CleanSpeak on Windows when it is not installed as a Windows Service, execute the startup.bat script located in the <cleanspeak-service>\apache-tomcat-7.0.40\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.

NOTE: If you are running these commands via a command.exe prompt, you MUST be in the <cleanspeak-service>\apache-tomcat-7.0.40\bin directory. Otherwise, the command will failed with an error stating that the CATALINA_HOME variable is not setup correctly.

Windows Service

If you installed CleanSpeak 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 CleanSpeak execute this command:

```
$ sudo /etc/init.d/<cleanspeak-service> start
```

To stop CleanSpeak execute this command:

```
$ sudo /etc/init.d/<cleanspeak-service> stop
```

Linux

To start CleanSpeak on Linux when it is not installed using the RPM or Debian packages, execute the startup.sh script located in the <cleanspeak-service>/apache-tomcat-7.0.40/bin directory like this:

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

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

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

Next Step: Testing and Monitoring

Testing and Monitoring

Testing and Monitoring CleanSpeak

This section describes how to test that CleanSpeak is running and provides recommended monitoring procedures.

Note: The port numbers for each service may need to be modified from the examples if the values were changed in the cleanspeak.properties configuration file.

Management Interface

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

http://localhost:8011/login

Performing a frequent GET request on the above URL is sufficient to monitor the Management Interface service.

Note: The Management Interface is not required to be running to filter content (the WebService is required).

WebService

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.

Performing a frequent GET request to the above URL will ensure that the WebService is running. In order to monitor filtering results from the WebService, it is recommended to send a POST request with an entry that should always be filtered. The body of the request will look like this:

```
contentItem.content=fuck&
contentItem.type=text&
filter.blacklist.enabled=true&
filter.operation=replace&
```

A complete cURL example:

```
curl -d
"contentItem.content=fuck&contentItem.type=text&filter.blacklist.enabled=t
rue&filter.operation=replace" http://localhost:8001/content/item.js
```

Note: You may need to add the authentication token to the header if Authentication is enabled.

The response (this example uses JSON):

```
{
  "filter": {
    "matched": true,
      "matches": [
        { "start": 0, "length": 4, "type": "blacklist", "locale": "en",
"matched": "fuck", "root": "fuck", "tags": ["Vulgarity"], "severity":
"severe", "quality": 1.00000 }
      ],
      "replacement": "****"
  }
}
```

Your monitoring solution should ensure that:

filter.matched = true

Note: When running multiple WebServices, we recommend monitoring each WebService individually as well as the load balancer if applicable. See Scaling the WebService for more on running multiple WebServices.

Search Engine

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

http://localhost:8021/

Performing a frequent GET request on the above URL is sufficient to monitor the Search Engine service. If multiple search engines are deployed, test each URL respectively.

Next Step: Scaling and Redundancy

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.

Next Step: Scaling the Management Interface

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.

Next Step: Scaling the Search Engine

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.

API

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	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 http://www.ietf.org/rfc/rfc2396.txt. 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: • http://localhost:8001/example/1 • http://localhost:8001/example?id=1 • http://localhost:8001/example?search=some+search+string
POST	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: http://www.w3.org/Protocols/rfc2616/rfc2616-sec4.html#sec4.3. The format of the message-body is specified by the HTTP Transfer-Encoding, which is specified by the **Content-Type** HTTP header. To use the message-body mechanism to send in the data you must set the **Content-Type** header to **application/x-www-form-urlencoded** . If you want more information on the **application/x-www-form-urlencoded** Transer-Encoding it is defined as part of the HTML specification here: http://www.w3.org/TR/html401/interact/forms.html. Here is an example of HTTP bodies for POST requests: • contentItem.content=testing&filter.enabled=true&filter.blacklist.enabled=true
PUT	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.
DELETE	Parameters for a DELETE request are are the same as the GET request. Similarly, the ID parameter can be set as a URL parameter or part of the URI.

Authentication

If the CleanSpeak WebService is publicly accessible, authentication **must** be enabled. Review the Authentication page for configuration and request parameters.

Filtering & Moderating Content

Filtering and storing content for moderation is performed with a POST request to the Content Item service.

Resource	Description
POST content/item	Send content for filtering and/or moderation.

User Attributes

Store details of the users in your application such as name, email, and age with the Content User service. Details for each user are displayed in the Management Interface within user profile pages.

Resource	Description
GET content/user	Returns a user's information.
POST content/user	Creates a new user.
PUT content/user	Updates an existing user.
DELETE content/user	Deletes an existing user.

System Users

Manage system users (moderators) with the System User service. The same actions can be performed within the Management Interface. See: System Users & Permissions.

Resource	Description
GET system/user	Retrieves a system user's information.

POST system/user	Creates a system user.
PUT system/user	Updates an existing system user.
DELETE system/user	Deletes an existing system user.

Next Step: Authentication

Authentication

Authentication Setup

If the WebService URL is publicly accessible, authentication **must** be enabled. Authentication is enabled by editing the cleanspeak.properties configuration file on the servers that the CleanSpeak WebService is installed on. You can read more about this configuration on the Configuration page of the documentation.

Request Authentication

Pass the authentication key you setup in the Management Interface (above) 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");
HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
urlConnection.addRequestProperty("Authentication", "1234-ABCD-5678-EFGH");
...
```

Here is a simple cURL sample:

```
curl -H "Authentication: 1234-ABCD-5678-EFGH" -d
"contentItem.content=testing&contentItem.type=text&filter.blacklist.enable
d=true&filter.operation=replace" http://localhost:8001/content/item.js
```

POST Content Item

The Content Item service provides a single API to perform all content handling functionality. POST requests are used for filtering, content storage & search indexing, and content moderation.

Based on the parameters you pass to the WebService and the configuration set from the Management Interface, CleanSpeak might perform one or more operations during each request.

Refer to the Concepts section of the user guide for an overview of integration ideas.

Request URLs

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

URL	Response Type
http://localhost:8001/content/item.js	JSON response
http://localhost:8001/content/item.xml	XML response
http://localhost:8001/content/item	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/or 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.
500	There was a CleanSpeak internal error. A stack trace is provided and logged in the Apache Tomcat /logs/catalina.out file.

The response body (JSON or XML) is detailed in the Response Reference section of the API documentation.

Parameters

General

You may send content to the CleanSpeak WebService individually or as a complex object that contains multiple content items of varying types. The following is a description for attributes to be set whether the content will be filtered, moderated, or both:

contentItem.type Required	The type of content as a *String*. Possible values: **text, hyperlink, image, video, complex** `contentItem.type=text`
contentItem.content Required for individual content Not allowed when contentItem.type=complex	*URL-Encoded String*. Descriptions for each content **type** (above): • **text:** The content to be filtered and/or moderated. Cannot exceed 65,000 characters. • **hyperlink:** Fully qualified URL to an external web page. • **image:** Fully qualified URL of the image to be displayed. • **video:** Fully qualified URL of the video to be displayed. `contentItem.content=Filter%20this%20message`
contentItem.complex[*].type Required for complex content Not allowed when contentItem.type!=complex	The type of content as a *String*. Possible values: **text, hyperlink, image, video, attribute** `contentItem.complex[0].type=hyperlink`
contentItem.complex[*].content Required for complex content Not allowed when contentItem.type!=complex	*URL-Encoded String*. Descriptions for each **type**: • **text:** The content to be filtered and/or moderated. Cannot exceed 65,000 characters. • **hyperlink:** Fully qualified URL to an external web page. • **image:** Fully qualified URL of the image to be displayed. • **video:** Fully qualified URL of the video to be displayed. • **attribute**: A free form attribute value that will not be filtered. `contentItem.complex[0].content=http://www.inversoft.com`
contentItem.complex[*].name Optional for complex content Not allowed when contentItem.type!=complex	An optional name for the complex content item as a *URL-Encoded String*. `contentItem.complex[0].name=Home%20Page`

Note: When passing complex objects, the values allowed inside the brackets are 0-N (integers), **and no index can be skipped.** The order of each parameter in the request is not important, but all the required elements must be present for each expected complex object. E.g. if you are passing 2 complex objects you can must specify content and type for indices 0 and 1, whereas passing indices 1 and 2 would fail.

Filtering

CleanSpeak can filter text (contentItem.type=text, or contentItem.complex[n].type=text) in order to find: Blacklist Entries, Characters, Email

Addresses, Phone Numbers, URLs, and Words. Each are a separate filter type and require the *filter.<type>.enabled=true* parameter.

Filtering blacklist entries in languages other than English requires importing the desired language blacklist. First, download all available lists from the Downloads section of our website (**Languages Database** in the Database Downloads section). Then, import the desired language lists from the Management Interface under Filter -> Import.

filter.operation depreciated	The operation to be performed as a *String*. This parameter should always be set to **replace**. Alternative parameters are **match** and **locate** which are both depreciated. *This parameter is no longer required.* `filter.operation=replace`
filter.blacklist.enabled required for blacklist filtering	Turns on the Blacklist filter as a *boolean*. `filter.blacklist.enabled=true`

filter.blacklist.tag
 	
optional
	This parameter controls which entries on your blacklist are filtered against based on the tags you have assigned to the entries as a *String*. You can specify multiple tags by including this parameter multiple times. By default, CleanSpeak filters against all tags. Tags are described in the List Management section of the user manual.
	
	`filter.blacklist.tag=Vulgarity&`
	`filter.blacklist.tag=Sexual`

filter.blacklist.locale
 	
optional
	This parameter controls which languages the filter will look for as a *String*. 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 certain locales. You can specify multiple locales by supplying this parameter multiple times. If left undefined, CleanSpeak will filter against all entries you have stored in your database.
	
	The format for locale is two digit lower case ISO 639-1 Language Code followed by an optional underscore and **upper case** two digit ISO 3166 country code.
	
	`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`

filter.blacklist.severity optional	This parameter controls which entries on your blacklist are filtered against based on the severity assigned to them as a *String*. 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 **high** the filter will search for words and phrases marked as **high** or **severe**. **Note**: The severity level needs to be all lower case. `filter.blacklist.severity=medium`
filter.characters.enabled required for single character filtering	Turns on the Characters filter as a *boolean*. `filter.characters.enabled=true`

filter.characters.character
 	
required for single character filtering
	This parameter specifies the individual characters that should be searched for as a *URL-Encoded char*. You can specify this parameter multiple times to filter multiple characters.
	
	`filter.characters.character=%40&`
	`filter.characters.character=%2A`

filter.emails.enabled required for email address filtering	Turns on the Email filter as a *boolean*. `filter.emails.enabled=true`

filter.emails.domainQuality.domain
 	
optional
	This parameter specifies the initial quality score for a specific domain as a *String*. For example, you can set the initial quality for the domain **it** to 0.5 by setting the **domain** paramet er to **it** and the **quality** 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:
	
	`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`

filter.emails.domainQuality.quality
 	
optional
	This parameter specifies the initial quality score for a specific domain as a *float*. For example, you can set the initial quality for the domain **it** to 0.5 by setting the **domain** parameter to **it** an d the **quality** 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.
	
	`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`

filter.emails.maximumMatchLength optional	This parameter controls the maximum length that a match can be in order to be considered an email as an *int*. This defaults to 50. `filter.emails.maximumMatchLength=60`
filter.emails.spacePenalty optional	This parameter specifies a penalty applied if the match contains any spaces as a *float*. For example, **foo@ bar. com** contains two spaces. This defaults to -0.05. `filter.emails.spacePenalty=-0.1`
filter.phoneNumbers.enabled required for phone number filtering	Turns on the Phone Number filter as a *boolean*. `filter.phoneNumbers.enabled=true`
filter.phoneNumbers.maximumMatchLength optional	This parameter specifies the maximum length that a match can be in order to be considered a phone number as an *int*. This defaults to 20 in order to cover most world wide phone number formats. `filter.phoneNumbers.maximumMatchLength=15`
filter.phoneNumbers.minimumMatchLength optional	This parameter specifies the minimum length that a match can be in order to be considered a phone number as an *int*. This defaults to 6 in order to cover most world wide phone number formats. `filter.phoneNumbers.minimumMatchLength=5`
filter.phoneNumbers.separatorPenalty optional	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 as a *float*. For example, **303;555;1234** contains two penalized separators. This defaults to -0.02. `filter.phoneNumbers.separatorPenalty=-0.1`
filter.phoneNumbers.spacePenalty optional	This parameter specifies a penalty that is applied to the quality score for a match if it contains a space as a *float*. For example, **303 555 1234** contains two spaces. This defaults to -0.02. `filter.phoneNumbers.spacePenalty=-0.1`
filter.phoneNumbers.wordPenalty optional	This parameter specifies a penalty that is applied to the quality score for a match if it contains any words rather than digits as a *float*. For example, **three zero three 555 1234** contains three words. This defaults to -0.03. `filter.phoneNumbers.wordPenalty=-0.1`

filter.urls.enabled required for URL address filtering	Turns on the URL filter as a *boolean*. `filter.urls.enabled=true`

filter.urls.domainQuality.domain
optional
	This parameter specifies the initial quality score for a specific domain as a *String*. For example, you can set the initial quality for the domain **it** to 0.5 by setting the **domain** parameter to **it** and the **quality** 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:
	`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`

filter.urls.domainQuality.quality
optional
	This parameter specifies the initial quality score for a specific domain as a *float*. For example, you can set the initial quality for the domain **it** to 0.5 by setting the **domain** parameter to **it** and the **quality** 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:
	`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`

filter.urls.maximumMatchLength optional	This parameter controls the maximum length that a match can be in order to be considered a URL as an *int*. This defaults to 50. `filter.urls.maximumMatchLength=60`
filter.urls.spacePenalty optional	This parameter specifies a penalty applied if the match contains any spaces as a *float*. This penalty is applied only once. For example, **w w w . ex a m ple . c o m** contains multiple spaces but is penalized once. This defaults to -0.05. `filter.urls.spacePenalty=-0.1`
filter.words.enabled required for specific word filtering	Turns on the Word filter as a *boolean*. `filter.words.enabled=true`

filter.words.word
required for specific word filtering
	This parameter specifies a word that the filter will search for. You can specify multiple words by supplying this parameter multiple times. For example:
	`filter.words.word=foo&`
	`filter.words.word=bar`

filter.replaceChar optional	The character that is used during a replace operation as a *URL-Encoded char*. The default replacement character is an asterisk. `filter.replaceChar=%24`
filter.replaceString optional	The String that is used during a replace operation as a *String*. `filter.replaceString=love`
filter.whitelist.enabled required for whitelist filtering	Turns on the Whitelist filter as a *boolean*. Can be used in conjunction with other filters. `filter.whitelist.enabled=true`

Storing Content & Moderation

Storing content requires configuring at least one Application and one Component in the Management Interface. See Moderation Setup & Configuration. Visit the Moderation Concepts section of the user guide for integration ideas and descriptions of prioritizing & queueing content.

The attribute **type** is a *URL-Encoded String* unless specified otherwise.

contentItem.application required for moderation	The name of the Application that the content was generated in. This must match the name you setup in the Management Interface exactly. `contentItem.application=My%20Game`
contentItem.component required for moderation	The name of the Component that the content was generated in. This must match the name you setup in the Management Interface exactly. `contentItem.component=Chat`
contentItem.createInstant required for moderation	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. `contentItem.createInstant=1361230448000`
contentItem.uid required for approving, editing, and deleting content	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. **NOTE**: This UID must be globally unique across all of your Applications and Components. `contentItem.uid=1234abcd`
contentItem.location optional	Specifies the audience of a message. For example, you might use an chat room name, area ID for a game, or a thread ID for a forum. This parameter is used by CleanSpeak to display conversational views of content with the Threaded View feature. **NOTE**: All one-to-one messages should have the same value for contentItem.location, such as "PrivateMessage" `contentItem.location=ForumThread121`
contentItem.senderUID optional	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. This value is required when integrating the User Actions feature. To add profile information about the user, review the Content User API. **NOTE**: This ID must be unique across all of your Applications and Components. `contentItem.senderUID=abc123`
contentItem.senderDisplayName optional	This is an optional parameter to display a different name in the management interface rather than the user UID such as character name or forum display name. The senderUID attribute must be set when using this parameter. A user UID may have multiple display names associated to it. `contentItem.senderDisplayName=CoolGuy`
contentItem.receiverUID optional	This is a unique identifier for the person that received the content. This should **only** be used for Private Messaging between two users. **NOTE**: This ID must be unique across all of your Applications and Components. `contentItem.receiverUID=abc123`
contentItem.receiverDisplayName optional	This is an optional parameter to display a different name in the management interface rather than the user UID such as character name or forum display name. The receiverUID attribute must be set when using this parameter. A user UID may have multiple display names associated to it. `contentItem.receiverDisplayName=CoolGuy`

moderation.type required for approving content	If sending content to an approval queue, this parameter must be set to the value **requiresApproval**. This parameter can also be set to **generatesAlert** to add the content to an alert queue. If filter rules are configured, and a filter rule is triggered by the content the moderation.type will be overridden by the filter rule's configured action. `moderation.type=requiresApproval`

See Also: Response Reference

GET Content User

The Content User service provides a single API to perform all user handling functionality. GET requests are used for retrieving user data that has been previously stored with a POST request.

- Request URLs
- Response Codes
- Parameters
- Sample Request

Request URLs

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

URL	Response Type
http://localhost:8001/content/user.js	JSON response
http://localhost:8001/content/user.xml	XML response
http://localhost:8001/content/user	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/or 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.
500	There was a CleanSpeak internal error. A stack trace is provided and logged in the Apache Tomcat /logs/catalina.out file.

Parameters

There is a single attribute to retrieve user data:

contentUser.uid required	The unique identifier of the user. `contentUser.uid=1234`

Sample Request

URL: http://localhost:8001/content/user.js

To retrieve user information from the POST example:

GET cURL request:

```
curl "http://localhost:8001/content/user.js?contentUser.uid=123456"
```

JSON Response:

```
{
  "id": "83340ba9-f3b9-483e-842b-9a376438c705",
  "uid": "123456",
  "name": "John Doe",
  "email": "john@example.com",
  "birthDateInstant": 631152000000,
  "attributes": {
    "Favorite Color": "blue",
    "Shoe Size": "10.5",
    "Transaction Total": "$25.50"
  },
  "displayNames": [
    "Character1","Character2"
  ]
}
```

XML Response (sent to http://localhost:8001/content/user):

```
<?xml version="1.0" encoding="UTF-8"?>
<user xmlns="http://www.inversoft.com/schemas/cleanspeak/content/user/2"
      id="83340ba9-f3b9-483e-842b-9a376438c705" uid="123456"
      name="John Doe"
      email="john@example.com"
      birthDateInstant="631152000000"
      >
    <attribute name="Favorite Color" value="blue"/>
    <attribute name="Shoe Size" value="10.5"/>
    <attribute name="Transaction Total" value="$25.50"/>
    <displayName name="Character1"/>
    <displayName name="Character2"/>
</user>
```

POST Content User

The Content User service provides a single API to perform all user handling functionality. POST requests are used for storing new users and should be integrated with your user registration/management system.

- Request URLs
- Response Codes
- Parameters
- Sample Request

Request URLs

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

URL	Response Type

http://localhost:8001/content/user.js	JSON response
http://localhost:8001/content/user.xml	XML response
http://localhost:8001/content/user	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/or 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.
500	There was a CleanSpeak internal error. A stack trace is provided and logged in the Apache Tomcat /logs/catalina.out file.

Parameters

The attribute **type** is a *URL-Encoded String* unless specified otherwise.

contentUser.uid required	The unique identifier of the user. **NOTE**: The Content User UID must must be unique across all of your configured Applications and Components. `contentUser.uid=1234`
contentUser.name optional	Full name of the user. `contentUser.name=John%20Doe`
contentUser.email optional	Email address of the user. `contentUser.email=john%40example.com`
contentUser.birthDateInstant optional	Birth date as a *long* value that represents the number of **milliseconds** since Epoch UTC: `contentUser.birthDateInstant=631213063000`
contentUser.lastLoginInstant optional	Time at last login as a *long* value that represents the number of **milliseconds** since Epoch UTC: `contentUser.birthDateInstant=1361811528000`

contentUser.displayNames
optional
	Display names for the user. This parameter can be specified multiple times for each display name.
	`contentUser.displayNames=Character1&`
	`contentUser.displayNames=Character2`

contentUser.attributes['name']
optional
	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:
	`contentUser.attributes['gender']=Male&`
	`contentUser.attributes['favorite-color']=Blue`

Sample Request

URL: http://localhost:8001/content/user.js

POST body:

```
contentUser.uid=123456&
contentUser.name=John%20Doe&
contentUser.email=john%40example.com&
contentUser.birthDateInstant=631213063000&
contentUser.displayNames=Character1&
contentUser.displayNames=Character2&
contentUser.attributes['Favorite%20Color']=blue&
contentUser.attributes['Shoe%20Size']=10.5&
contentUser.attributes['Transaction%20Total']=%2425.50
```

Complete cURL request:

```
curl -d
"contentUser.uid=123456&contentUser.name=John%20Doe&contentUser.email=john
%40example.com&contentUser.birthDateInstant=631213063000&contentUser.displ
ayNames=Character1&contentUser.displayNames=Character2&contentUser.attribu
tes['Favorite%20Color']=blue&contentUser.attributes['Shoe%20Size']=10.5&co
ntentUser.attributes['Transaction%20Total']=%2425.50"
http://localhost:8001/content/user.js
```

PUT Content User

The Content User service provides a single API to perform all user handling functionality. PUT requests are used when profile information is updated in your user registration/management system. The user must be stored first with a POST request prior to updating with a PUT request.

NOTE: When updating a user, all existing attributes will be removed and the new attributes will be stored. Be sure to add existing (unmodified) data to content user PUT requests if it they should remain in CleanSpeak.

- Request URLs
- Response Codes
- Parameters
- Sample Request

Request URLs

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

URL	Response Type
http://localhost:8001/content/user.js	JSON response
http://localhost:8001/content/user.xml	XML response
http://localhost:8001/content/user	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/or 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.
500	There was a CleanSpeak internal error. A stack trace is provided and logged in the Apache Tomcat /logs/catalina.out file.

Parameters

The attribute **type** is a *URL-Encoded String* unless specified otherwise.

contentUser.uid required	The unique identifier of the user as originally sent from the POST request. **NOTE**: The Content User UID must must be unique across all of your configured Applications and Components. `contentUser.uid=1234`
contentUser.name optional	Full name of the user. `contentUser.name=John%20Doe`
contentUser.email optional	Email address of the user. `contentUser.email=john%40example.com`
contentUser.birthDateInstant optional	Birth date as a *long* value that represents the number of **milliseconds** since Epoch UTC: `contentUser.birthDateInstant=631213063000`
contentUser.lastLoginInstant optional	Time at last login as a *long* value that represents the number of **milliseconds** since Epoch UTC: `contentUser.birthDateInstant=1361811528000`

contentUser.displayNames
 	
optional
	Display names for the user. This parameter can be specified multiple times for each display name.
	
	`contentUser.displayNames=Character1&`
	`contentUser.displayNames=Character2`

contentUser.attributes['name']
 	
optional
	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:
	
	`contentUser.attributes['gender']=Male&`
	`contentUser.attributes['favorite-color']=Blue`

Sample Request

URL: http://localhost:8001/content/user.js

Original POST body:

```
contentUser.uid=123456&
contentUser.name=John%20Doe&
contentUser.email=john%40example.com&
contentUser.birthDateInstant=631213063000&
contentUser.displayNames=Character1&
contentUser.displayNames=Character2&
contentUser.attributes['Favorite%20Color']=blue&
contentUser.attributes['Shoe%20Size']=10.5&
contentUser.attributes['Transaction%20Total']=%2425.50
```

PUT body to update the Transaction Total:

```
contentUser.uid=123456&
contentUser.name=John%20Doe&
contentUser.email=john%40example.com&
contentUser.birthDateInstant=631213063000&
contentUser.displayNames=Character1&
contentUser.displayNames=Character2&
contentUser.attributes['Favorite%20Color']=blue&
contentUser.attributes['Shoe%20Size']=10.5&
contentUser.attributes['Transaction%20Total']=%2432.00
```

Complete original POST cURL request:

```
curl -d
"contentUser.uid=123456&contentUser.name=John%20Doe&contentUser.email=john
%40example.com&contentUser.birthDateInstant=631213063000&contentUser.displ
ayNames=Character1&contentUser.displayNames=Character2&contentUser.attribu
tes['Favorite%20Color']=blue&contentUser.attributes['Shoe%20Size']=10.5&co
ntentUser.attributes['Transaction%20Total']=%2425.50"
http://localhost:8001/content/user.js
```

Complete update PUT cURL request:

```
curl -X PUT -d
"contentUser.uid=123456&contentUser.name=John%20Doe&contentUser.email=john
%40example.com&contentUser.birthDateInstant=631213063000&contentUser.displ
ayNames=Character1&contentUser.displayNames=Character2&contentUser.attribu
tes['Favorite%20Color']=blue&contentUser.attributes['Shoe%20Size']=10.5&co
ntentUser.attributes['Transaction%20Total']=%2432.00"
http://localhost:8001/content/user.js
```

DELETE Content User

The Content User service provides a single API to perform all user handling functionality. DELETE requests are used for deleting user data that has been previously stored with a POST request.

Request URLs

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

URL	Response Type
http://localhost:8001/content/user.js	JSON response
http://localhost:8001/content/user.xml	XML response
http://localhost:8001/content/user	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/or 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.
500	There was a CleanSpeak internal error. A stack trace is provided and logged in the Apache Tomcat /logs/catalina.out file.

Parameters

There is a single attribute to delete user data:

contentUser.uid required	The unique identifier of the user. `contentUser.uid=1234`

Sample Request

URL: http://localhost:8001/content/user.js

To delete the user from the POST example:

DELETE cURL request:

```
curl -X DELETE "http://localhost:8001/content/user?contentUser.uid=123456"
```

GET System User

The System User service provides a single API to perform all system user handling functionality. System users are referred to as Moderators in the CleanSpeak Management Interface (System -> Moderators). GET requests are used to retrieve details about a system user.

- Request URLs
- Response Codes
- Parameters
- Sample Request

Request URLs

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

URL	Response Type
http://localhost:8001/system/user.js	JSON response
http://localhost:8001/system/user.xml	XML response
http://localhost:8001/system/user	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/or 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.
500	There was a CleanSpeak internal error. A stack trace is provided and logged in the Apache Tomcat /logs/catalina.out file.

Parameters

There is a single attribute to retrieve system user data:

systemUser.uid required	The unique identifier of the user. `systemUser.uid=1`

Sample Request

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

To retrieve system user information from the POST example:

GET cURL request:

```
curl "http://localhost:8001/system/user.js?systemUser.id=6"
```

JSON Response:

```
{
  "id": 6,
  "email": "example@inversoft.com",
  "roles": [
    {"id": 8, "name": "approve_all"},
    {"id": 4, "name": "moderator"}
  ],
  "applications": [
    {
      "id": 2,
      "name": "Adult Game",
      "components": [
        {"id": 3, "name": "Chat"}
      ]
    }
  ]
}
```

XML Response (sent to http://localhost:8001/system/user):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<user xmlns="http://www.inversoft.com/schemas/cleanspeak/system/user/2"
      id="6" email="example@inversoft.com">
  <role id="8" name="approve_all"/>
  <role id="4" name="moderator"/>
  <application id="2" name="Adult Game">
    <component id="3" name="Chat"/>
  </application>
</user>
```

POST System User

The System User service provides a single API to perform all system user handling functionality. System users are referred to as Moderators in the CleanSpeak Management Interface (System -> Moderators). POST requests are used for storing new system users.

Request URLs

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

URL	Response Type
http://localhost:8001/system/user.js	JSON response
http://localhost:8001/system/user.xml	XML response
http://localhost:8001/system/user	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/or 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.
500	There was a CleanSpeak internal error. A stack trace is provided and logged in the Apache Tomcat /logs/catalina.out file.

Parameters

The attribute **type** is a *URL-Encoded String* unless specified otherwise.

systemUser.email required	The email (username) for the user. `systemUser.email=example%40inversoft.com`
systemUser.password required	The password for the user. `systemUser.password=password`

systemUser.role
 	
optional
	The role(s) for the user. You can specify this parameter multiple times, once for each role. Available roles are:
	
	• admin
	• approve_all
	• audit_viewer
	• filter
	• filter_manager
	• moderator
	• moderator_manager
	• user_admin
	
	`systemUser.role=moderator&`
	`systemUser.role=approve_all`
	
	See also: System Users & Permissions

systemUser.applicationID
 	
optional
	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. In order to retrieve the primary keys from the CleanSpeak database, log into your database and perform a select on the applications table:
	
	`mysql> select * from applications;`
	
	`systemUser.applicationID=2&`
	`systemUser.applicationID=3`
	
	See also: System Users & Permissions

Sample Request

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

POST body:

```
systemUser.email=example%40inversoft.com&
systemUser.password=password&
systemUser.role=moderator&
systemUser.role=approve_all&
systemUser.applicationID=1
```

Complete cURL request:

```
curl -d
"systemUser.email=example%40inversoft.com&systemUser.password=password&sys
temUser.role=moderator&systemUser.role=approve_all&systemUser.applicationI
D=1" http://localhost:8001/system/user.js
```

JSON Response:

```
{
  "id": 6,
  "email": "example@inversoft.com",
  "roles": [
    {"id": 4, "name": "moderator"},
    {"id": 8, "name": "approve_all"}
  ],
  "applications": [
    {
      "id": 2,
      "name": "Adult Game",
      "components": [
        {"id": 3, "name": "Chat"}
      ]
    }
  ]
}
```

XML Response (sent to http://localhost:8001/system/user):

```
<?xml version="1.0" encoding="UTF-8"?>
<user xmlns="http://www.inversoft.com/schemas/cleanspeak/system/user/2"
      id="6" email="example@inversoft.com">
  <role id="4" name="moderator"/>
  <role id="8" name="approve_all"/>
  <application id="2" name="Adult Game">
    <component id="3" name="Chat"/>
  </application>
</user>
```

See Also:

Response Reference

Integration Examples

PUT System User

The System User service provides a single API to perform all system user handling functionality. System users are referred to as Moderators in the CleanSpeak Management Interface (System -> Moderators). PUT requests are used updating system users. The system user must be stored first with a POST request prior to updating with a PUT request.

- Request URLs
- Response Codes
- Parameters
- Sample Request

Request URLs

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

URL	Response Type
http://localhost:8001/system/user.js	JSON response
http://localhost:8001/system/user.xml	XML response

http://localhost:8001/system/user	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/or 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.
500	There was a CleanSpeak internal error. A stack trace is provided and logged in the Apache Tomcat /logs/catalina.out file.

Parameters

The attribute **type** is a *URL-Encoded String* unless specified otherwise.

systemUser.id
 	
required
	This specifies the user to update. System user IDs can be retrieved from the database with the following command:
	
	`mysql> select * from users;`
	
	`systemUser.id=3`

systemUser.email optional	The email (username) for the user. `systemUser.email=example%40inversoft.com`
systemUser.password optional	The password for the user. If not set, the password will be unchanged. `systemUser.password=new_password`

systemUser.role
 	
optional
	The role(s) for the user. You can specify this parameter multiple times, once for each role. Available roles are:
	
	• admin
	• approve_all
	• audit_viewer
	• filter
	• filter_manager
	• moderator
	• moderator_manager
	• user_admin
	
	`systemUser.role=moderator&`
	`systemUser.role=approve_all`
	
	See also: System Users & Permissions

systemUser.applicationID
 	
optional
	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. In order to retrieve the primary keys from the CleanSpeak database, log into your database and perform a select on the applications table:
	
	`mysql> select * from applications;`
	
	`systemUser.applicationID=2&`
	`systemUser.applicationID=3`
	
	See also: System Users & Permissions

Sample Request

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

Original POST body:

```
systemUser.email=example%40inversoft.com&
systemUser.password=password&
systemUser.role=moderator&
systemUser.role=approve_all&
systemUser.applicationID=1
```

PUT body to update the role to include filter_manager:

```
systemUser.email=example%40inversoft.com&
systemUser.role=moderator&
systemUser.role=approve_all&
systemUser.role=filter_manager&
systemUser.applicationID=1&
systemUser.id=6
```

Complete original POST cURL request:

```
curl -d
"systemUser.email=example%40inversoft.com&systemUser.password=password&sys
temUser.role=moderator&systemUser.role=approve_all&systemUser.applicationI
D=1" http://localhost:8001/system/user.js
```

Complete update PUT cURL request:

```
curl -X PUT -d
"systemUser.email=example%40inversoft.com&systemUser.role=moderator&system
User.role=approve_all&systemUser.role=filter_manager&systemUser.applicatio
nID=1&systemUser.id=6" http://localhost:8001/system/user.js
```

JSON Response:

```
{
  "id": 6,
  "email": "example@inversoft.com",
  "roles": [
    {"id": 8, "name": "approve_all"},
    {"id": 3, "name": "filter_manager"},
    {"id": 4, "name": "moderator"}
  ],
  "applications": [
    {
      "id": 1,
      "name": "Game",
      "components": [
        {"id": 1, "name": "Chat"},
        {"id": 2, "name": "Forums"}
      ]
    }
  ]
}
```

XML Response (sent to http://localhost:8001/system/user):

```
<?xml version="1.0" encoding="UTF-8"?>
<user xmlns="http://www.inversoft.com/schemas/cleanspeak/system/user/2"
      id="6" email="example@inversoft.com">
  <role id="3" name="filter_manager"/>
  <role id="4" name="moderator"/>
  <role id="8" name="approve_all"/>
  <application id="1" name="Game">
    <component id="1" name="Chat"/>
    <component id="2" name="Forums"/>
  </application>
</user>
```

DELETE System User

The System User service provides a single API to perform all system user handling functionality. System users are referred to as Moderators in the CleanSpeak Management Interface (System -> Moderators). DELETE requests are used to delete a system user.

- Request URLs
- Response Codes
- Parameters
- Sample Request

Request URLs

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

URL	Response Type
http://localhost:8001/system/user.js	JSON response
http://localhost:8001/system/user.xml	XML response

http://localhost:8001/system/user	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/or 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.
500	There was a CleanSpeak internal error. A stack trace is provided and logged in the Apache Tomcat /logs/catalina.out file.

Parameters

There is a single attribute to delete a system user:

systemUser.uid required	The unique identifier of the user. `systemUser.uid=1`

Sample Request

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

To retrieve system user information from the POST example:

GET cURL request:

```
curl -X DELETE "http://localhost:8001/system/user.js?systemUser.id=6"
```

Response Reference

- Content Item - JSON
    - Sample JSON Request/Response - Individual Content Item
    - Sample JSON Request/Response - Complex Content Item (Queue for Approval)
- Content Item - XML Schema
    - Sample XML Request/Response - Individual Content Item
    - Sample XML Request/Response - Complex Content Item (Queue for Approval)

Whitelist

The tables below describe each element of the response object when sending content via POST requests to the content item service.

Content Item - JSON

JSON Key	Applicable to complex response	Value Description

contentAction	Yes	The action that should be taken on the content as configured in the Management Interface. This is only applicable when the CleanSpeak moderation platform is licensed and the request specifies the application name, component name, and create instant (contentItem.application, contentItem.component, and contentItem.createInstant). Possible values: **allow**, **authorOnly**, **replace**, **queuedForApproval**, and **reject**
filter	Yes	The filter object that contains the result from a filter request.
filter.matched	Yes	A boolean that defines if the filter found any results or not.
filter.matches	No	A list of matches that the filter found.
filter.matches.start	No	The start index of a single match.
filter.matches.length	No	The length of a single match.
filter.matches.type	No	The type of the match. This will correspond to the filter types that were enabled and configured in the request.
filter.matches.locale	No	The locale of a single match.
filter.matches.matched	No	If the match is of the type **blacklist** or **whitelist**, this will contain the value that the filter matched on, which may be an inflection or a variation of a blacklist entry.
filter.matches.root	No	If the match is of the type **blacklist** or a **whitelist disallowedPhrase**, this will contain the root word/phrase of the match.
filter.matches.tags	No	If the match is of the type **blacklist**, this will contain an array with all of the tags of the blacklist entry that was matched.
filter.matches.severity	No	If the match is of the type **blacklist**, this will contain the severity of the match.
filter.matches.quality	No	The quality score of the match. Matches of type **blacklist**, **characters**, or **words** will always return with a value of 1.0. Matches of type **emails**, **phoneNumbers**, and **urls** will return with a value between 0.0 - 1.0.
filter.replacement	No	The content with all matches replaced.
moderation	Yes	The moderation object that contains the result from a Moderation request.
moderation.type	Yes	The type of moderation: **requiresApproval**, **requiresReview**, **generatesAlert**, or **unmoderated**. If **unmoderated**, the content has been been stored but not queued for moderator review.
moderation.queued	Yes	A boolean that indicates if the content was added to a queue for moderator review.
moderation.stored	Yes	A boolean that indicates if the content was stored or not.
moderation.alertsEnabled	Yes	A boolean that indicates if alerts are enabled for the application and component specified in the request.
moderation.approvalsEnabled	Yes	A boolean that indicates if approvals are enabled for the application and component specified in the request.
moderation.alertTags	Yes	If the content generated an alert, this will contain an array with all of the tags that the alert was generated by.
complex	Yes	A list of complex responses
complex[n].name	Yes	The name of the nth complex item from a filter request.
complex[n].filter	Yes	The filter object that contains the result for the nth complex item from a filter request.
complex[n].filter.matched	Yes	A boolean that defines if the filter for the nth complex item found any results or not.
complex[n].filter.matches	Yes	A list of matches that the filter found for the nth complex item.
complex[n].filter.matches.start	Yes	The start index of a single match for the nth complex item.
complex[n].filter.matches.length	Yes	The length of a single match for the nth complex item.
complex[n].filter.matches.type	Yes	The type of the match for the nth complex item. This will correspond to the filter types that were enabled and configured in the request.

complex[n].filter.matches.locale	Yes	The locale of a single match for the nth complex item.
complex[n].filter.matches.matched	Yes	If the match is of the type **blacklist** or **whitelist**, this will contain the value that the filter matched on for the nth complex item, which may be an inflection or a variation of a blacklist entry.
complex[n].filter.matches.root	Yes	If the match is of the type **blacklist** or a **whitelist disallowedPhrase**, this will contain the root word/phrase of the match for the nth complex item.
complex[n].filter.matches.tags	Yes	If the match is of the type **blacklist**, this will contain an array with all of the tags of the blacklist entry that was matched for the nth complex item.
complex[n].filter.matches.severity	Yes	If the match is of the type **blacklist**, this will contain the severity of the match for the nth complex item.
complex[n].filter.matches.quality	Yes	The quality score of the match for the nth complex item. Matches of type **blacklist**, **characters**, or **words** will always return with a value of 1.0. Matches of type **emails**, **phoneNumbers**, and **urls** will return with a value between 0.0 - 1.0.
complex[n].filter.replacement	Yes	The content with all matches replaced for the nth complex item.

Sample JSON Request/Response - Individual Content Item

URL: http://localhost:8001/content/item.js

POST body:

```
contentItem.type=text&
contentItem.content=what%20your%20pass&
filter.operation=replace&
filter.blacklist.enabled=true&
contentItem.application=MyGame&
contentItem.component=Chat&
contentItem.createInstant=1361230448000
```

Complete cURL request:

```
curl -d
"contentItem.content=what%20your%20pass&contentItem.type=text&filter.opera
tion=replace&filter.blacklist.enabled=true&contentItem.application=MyGame&
contentItem.component=Chat&contentItem.createInstant=1361230448000"
http://localhost:8001/content/item.js
```

With the Chat Component setup to generate alerts with a filter rule of authorOnly, here is the response body:

```
{
  "contentAction" : "authorOnly",
  "filter": {
    "matched": true,
      "matches": [
        { "start": 0, "length": 14, "type": "blacklist", "locale": "en",
"matched": "what your pass", "root": "what your pass", "tags":
["Account","PII"], "severity": "severe", "quality": 1.00000 }
      ],
      "replacement": "**************"
  },
  "moderation": {
    "type": "generatesAlert",
    "queued": true,
    "stored": true
  }
}
```

Sample JSON Request/Response - Complex Content Item (Queue for Approval)

URL: http://localhost:8001/content/item.js

POST body:

```
contentItem.type=complex&
contentItem.complex[0].type=text&
contentItem.complex[0].content=My%20First%20Post&
contentItem.complex[0].name=Title&
contentItem.complex[1].type=text&
contentItem.complex[1].content=I%20hope%20you%20enjoy%20my%20first%20YouTu
be%20video&
contentItem.complex[1].name=Body&
contentItem.complex[2].type=video&
contentItem.complex[2].content=http://www.youtube.com/watch%3Fv=KPsUveWBgY
M&
contentItem.complex[2].name=YouTubeVideo1&
filter.operation=replace&
filter.blacklist.enabled=true&
contentItem.application=MyGame&
contentItem.component=Forum&
contentItem.createInstant=1368047913000&
contentItem.uid=abcd1234&
moderation.type=requiresApproval
```

Complete cURL request:

```
curl -d
"contentItem.type=complex&contentItem.complex[0].type=text&contentItem.com
plex[0].content=My%20First%20Post&contentItem.complex[0].name=Title&conten
tItem.complex[1].type=text&contentItem.complex[1].content=I%20hope%20you%2
0enjoy%20my%20first%20YouTube%20video&contentItem.complex[1].name=Body&con
tentItem.complex[2].type=video&contentItem.complex[2].content=http://www.y
outube.com/watch%3Fv=KPsUveWBgYM&contentItem.complex[2].name=YouTubeVideo1
&filter.operation=replace&filter.blacklist.enabled=true&contentItem.applic
ation=MyGame&contentItem.component=Forum&contentItem.createInstant=1368047
913000&contentItem.uid=abcd1234&moderation.type=requiresApproval"
http://localhost:8001/content/item.js
```

Here is the response body:

```
{
  "filter": {
    "matched": true  },
  "moderation": {
    "type": "requiresApproval",
    "queued": true,
    "stored": true
  },
  "complex": [
      {
        "name": "Title",
          "filter": {
            "matched": false,
            "matches": [
            ],
            "replacement": "My First Post"
          }
      },
      {
        "name": "Body",
          "filter": {
            "matched": true,
            "matches": [
              { "start": 26, "length": 7, "type": "blacklist", "locale":
"en", "matched": "youtube", "root": "you tube", "tags": ["Grooming","PII"],
"severity": "medium", "quality": 1.00000 }
            ],
            "replacement": "I hope you enjoy my first ******* video"
          }
      },
      {
        "name": "YouTubeVideo1"
      }
  ]
}
```

Content Item - XML Schema

There is only one schema that validates both simple and complex type XML responses, and it is included with your Cleanspeak download (item.xsd).

Sample XML Request/Response - Individual Content Item

URL: http://localhost:8001/content/item.xml

POST body:

```
contentItem.type=text&
contentItem.content=what%20your%20pass&
filter.operation=replace&
filter.blacklist.enabled=true&
contentItem.application=MyGame&
contentItem.component=Chat&
contentItem.createInstant=1361230448000
```

Complete cURL request:

```
curl -d
"contentItem.content=what%20your%20pass&contentItem.type=text&filter.opera
tion=replace&filter.blacklist.enabled=true&contentItem.application=MyGame&
contentItem.component=Chat&contentItem.createInstant=1361230448000"
http://localhost:8001/content/item.xml
```

With the Chat Component setup to generate alerts, here is the response body:

```
<?xml version="1.0" encoding="UTF-8"?>
<handle xmlns="http://www.inversoft.com/schemas/cleanspeak/content/item/2">
    <contentAction action="authorOnly"/>
      <filter matched="true">
        <match start="0" length="14" type="blacklist" locale="en"
matched="what your pass" root="what your pass" severity="severe"
quality="1.00000">
          <tag>Account</tag>
          <tag>PII</tag>
        </match>
        <replacement><![CDATA[**************]]></replacement>
      </filter>
    <moderation type="generatesAlert" queued="true" stored="true"/>
</handle>
```

Sample XML Request/Response - Complex Content Item (Queue for Approval)

URL: http://localhost:8001/content/item.xml

POST body:

```
contentItem.type=complex&
contentItem.complex[0].type=text&
contentItem.complex[0].content=My%20First%20Post&
contentItem.complex[0].name=Title&
contentItem.complex[1].type=text&
contentItem.complex[1].content=I%20hope%20you%20enjoy%20my%20first%20YouTu
be%20video&
contentItem.complex[1].name=Body&
contentItem.complex[2].type=video&
contentItem.complex[2].content=http://www.youtube.com/watch%3Fv=KPsUveWBgY
M&
contentItem.complex[2].name=YouTubeVideo1&
filter.operation=replace&
filter.blacklist.enabled=true&
contentItem.application=MyGame&
contentItem.component=Forum&
contentItem.createInstant=1368047913000&
contentItem.uid=abcd1234&
moderation.type=requiresApproval
```

Complete cURL request:

```
curl -d
"contentItem.type=complex&contentItem.complex[0].type=text&contentItem.com
plex[0].content=My%20First%20Post&contentItem.complex[0].name=Title&conten
tItem.complex[1].type=text&contentItem.complex[1].content=I%20hope%20you%2
0enjoy%20my%20first%20YouTube%20video&contentItem.complex[1].name=Body&con
tentItem.complex[2].type=video&contentItem.complex[2].content=http://www.y
outube.com/watch%3Fv=KPsUveWBgYM&contentItem.complex[2].name=YouTubeVideo1
&filter.operation=replace&filter.blacklist.enabled=true&contentItem.applic
ation=MyGame&contentItem.component=Forum&contentItem.createInstant=1368047
913000&contentItem.uid=abcd1234&moderation.type=requiresApproval"
http://localhost:8001/content/item.xml
```

With the Chat Component setup to generate alerts, here is the response body:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<handle xmlns="http://www.inversoft.com/schemas/cleanspeak/content/item/2">
      <filter matched="true" />
    <moderation type="requiresApproval" queued="true" stored="true"/>
    <complex name="Title">
        <filter matched="false">
            <replacement><![CDATA[My First Post]]></replacement>
        </filter>
    </complex>
    <complex name="Body">
        <filter matched="true">
            <match start="26" length="7" type="blacklist" locale="en"
matched="youtube" root="you tube" severity="medium" quality="1.00000">
                  <tag>Grooming</tag>
                  <tag>PII</tag>
            </match>
            <replacement><![CDATA[I hope you enjoy my first *******
video]]></replacement>
        </filter>
    </complex>
    <complex name="YouTubeVideo1">
    </complex>
</handle>
```

Whitelist

This request hits five allowed words (my, first, post, bad, and phrase), one unrecognized word (jibberish), and one disallowed phrase (bad phrase).

URL: http://localhost:8001/content/item.xml

POST body:

```
contentItem.type=complex&
contentItem.complex[0].type=text&
contentItem.complex[0].content=My%20First%20Post%20jibberish%20bad%20phras
e&
contentItem.complex[0].name=test&
filter.operation=replace&
filter.whitelist.enabled=true
```

Complete cURL request:

```
curl -d
"contentItem.type=complex&contentItem.complex[0].type=text&contentItem.com
plex[0].content=My%20First%20Post%20jibberish%20bad%20phrase&contentItem.c
omplex[0].name=test&filter.operation=replace&filter.whitelist.enabled=true"
http://localhost:8001/content/item.xml
```

With the Chat Component setup to generate alerts, here is the response body (followed by the JSON response for the same request):

```
XML:
<?xml version="1.0" encoding="UTF-8"?>
<handle xmlns="http://www.inversoft.com/schemas/cleanspeak/content/item/2">
    <filter matched="true"/>
    <complex name="test">
        <filter matched="true">
              <match start="14" length="9" type="whitelist"
whitelistResult="disallowedWord" locale="" matched="jibberish" root=""
quality="1.00000">
              </match>
              <match start="24" length="10" type="whitelist"
whitelistResult="disallowedPhrase" locale="en" matched="bad phrase"
root="bad phrase" quality="1.00000">
              </match>
            <replacement><![CDATA[My First Post *********
**********]]></replacement>
        </filter>
    </complex>
</handle>

JSON:
{
  "filter": {
      "matched": true
  },

  "complex": [
      {
        "name": "test",
          "filter": {
            "matched": true,
            "matches": [
                  { "start": 14,
                    "length": 9,
                    "type": "whitelist",
                    "whitelistResult": "disallowedWord",
                    "locale": "",
                    "matched": "jibberish",
                    "root": "",
                    "quality": 1.00000 },
                  { "start": 24,
                    "length": 10,
                    "type": "whitelist",
                    "whitelistResult": "disallowedPhrase",
                    "locale": "en",
                    "matched": "bad phrase",
                    "root": "bad phrase",
                    "quality": 1.00000 }
            ],
              "replacement": "My First Post ********* **********"
          }
        }
    ]
}
```

Examples

This document covers how to quickly get started using the CleanSpeak WebService with Java and PHP. Detailed integration examples (using cURL requests) can be reviewed with the following links:

- Filtering without Moderation
- Filter Controls
- Images & Video
- Username Filtering & Moderation
- User Registration
- Taking Actions on Users
- System Users

To get started using the WebService, you will need to send an HTTP request with the correct parameters. We'll cover brief RESTful examples using Java and two simple filter requests 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 how to access a RESTful WebService. These examples are all in Java, but other languages have similar APIs.

GET Example

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

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

Java Filter Operation with an XML response

This example code illustrates a filter operation with a POST request to the content item service in Java that returns XML results.

```
URL url = new URL("http://localhost:8001/content/item.xml");
HttpURLConnection urlConnection = (HttpURLConnection) 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(inputStrea
m);
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 Filter Operation with an XML response

This example code illustrates a filter operation with a POST request to the content item service 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");
  }
}
```

Filtering without Moderation

Use Case

This section provides details to filter content without moderation or storage. For this example, CleanSpeak will filter against a single blacklist Tag *Vulgarity* and filter email addresses.

Configuring CleanSpeak

Filtering content without storage/moderation does not require any configuration within the management interface prior to code integration.

Sending the Request

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 API section.

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
http://localhost:8001/content/item.js	JSON response
http://localhost:8001/content/item.xml	XML response
http://localhost:8001/content/item	XML response

HTTP Method

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

Request Body

Request Attribute	Required?	Description
contentItem.content=<URL Encoded content>	**Yes**	The content to be filtered.
contentItem.type=text	**Yes**	For filter requests, the value for type must be **text**.
filter.blacklist.enabled=true	**Yes**	This enables the blacklist filter.
filter.blacklist.tag=Vulgarity	No	Tells CleanSpeak to only filter against entries tagged as *Vulgarity* only.
filter.emails.enabled=true	No	Tells CleanSpeak to filter email addresses.
filter.operation=replace	**Yes**	The operation used should always be **replace**.

NOTE: Review the API POST Content Item page for details and options for each request parameter.

Handling the 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/or 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.
500	There was a CleanSpeak internal error. A stack trace is provided and logged in the Apache Tomcat /logs/catalina.out file.

Sample cURL Request

Content: This message contains fuck as a profane word and support at inversoft dot com as an email

```
curl -X POST -d
"contentItem.content=This+message+contains+fuck+as+a+profane+word+and+supp
ort+at+inversoft+dot+com+as+an+email&contentItem.type=text&filter.blacklis
t.enabled=true&filter.blacklist.tag=Vulgarity&filter.emails.enabled=true&f
ilter.operation=replace" http://localhost:8001/content/item.js
```

JSON Response

```
{
  "filter": {
    "matched": true,
    "matches": [
      { "start": 22, "length": 4, "type": "blacklist", "locale": "en",
"matched": "fuck", "root": "fuck", "tags": ["Vulgarity"], "severity":
"severe", "quality": 1.00000 },
      { "start": 49, "length": 28, "type": "emails", "quality": 0.90000 }
    ],
    "replacement": "This message contains **** as a profane word and
**************************** as an email"
  }
}
```

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.
filter.matches.start	The start index of a single match.
filter.matches.length	The length 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.
filter.matches.locale	The locale of a single match.
filter.matches.matched	If the match is of the type **blacklist**, this will contain the value that the filter matched on, which may be an inflection or a variation of a blacklist entry.
filter.matches.root	If the match is of the type **blacklist**, this will contain the root word of the match.
filter.matches.tags	If the match is of the type **blacklist**, this will contain an array with all of the tags of the blacklist entry that was matched.
filter.matches.severity	If the match is of the type **blacklist**, this will contain the severity of the match.
filter.matches.quality	The quality score of the match. Matches of type **blacklist**, **characters**, or **words** will always return with a value of 1.0. Matches of type **emails**, **phoneNumbers**, and **urls** will return with a value between 0.0 - 1.0.
filter.replacement	The content with all matches replaced.

XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<handle xmlns="http://www.inversoft.com/schemas/cleanspeak/content/item/2">
  <filter matched="true">
    <match start="22" length="4" type="blacklist" locale="en"
matched="fuck" root="fuck" severity="severe" quality="1.00000">
      <tag>Vulgarity</tag>
    </match>
    <match start="49" length="28" type="emails" quality="0.90000">
    </match>
    <replacement><![CDATA[This message contains **** as a profane word and
**************************** as an email]]></replacement>
  </filter>
</handle>
```

Filter Controls

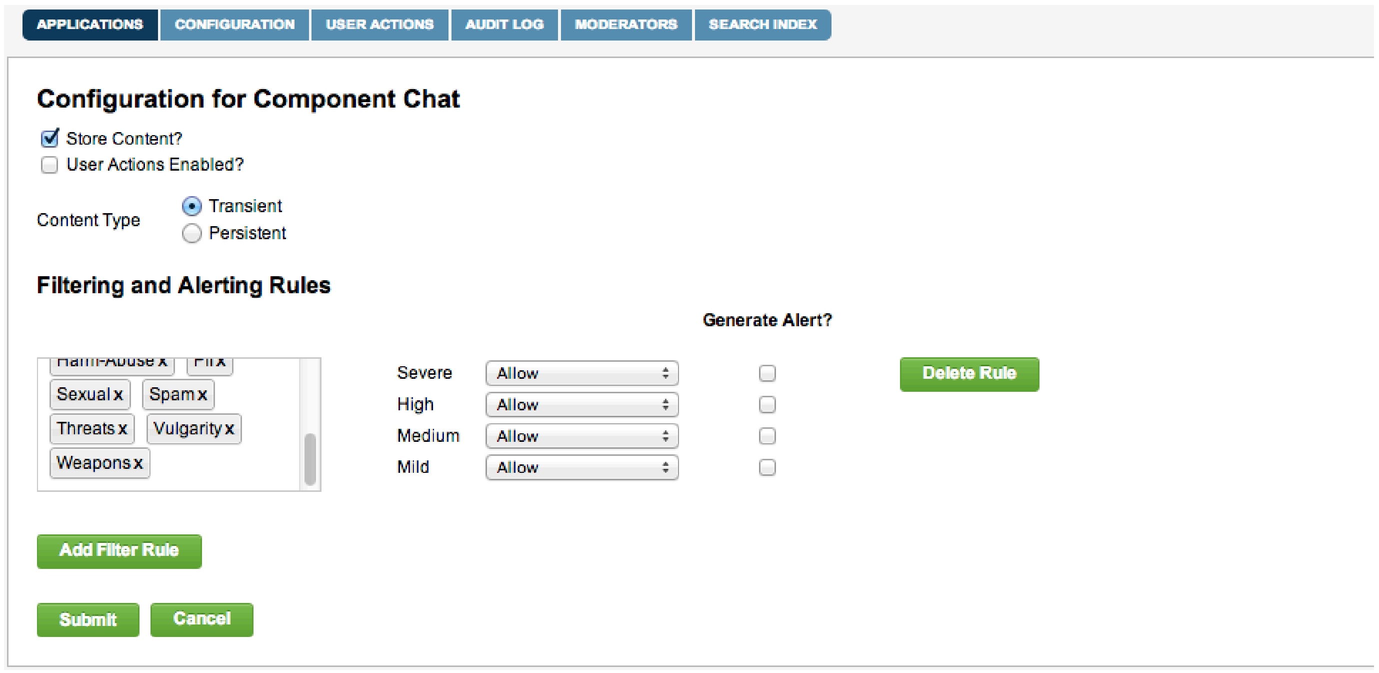
Use Case

This section provides details to filter content depending on the filter reaction configuration set in the Management Interface. For this example, CleanSpeak will return instructions to **reject** the content when a *severe* or *high* severity match is found and display the message to the **author only** when a *medium* or *mild* severity match is found. For brevity, this example will use a JSON request URL and response. Review the Filtering without Moderation example to review an XML example.

Configuring CleanSpeak

A component must be configured to return the proper filter response (contentAction) based on the severity of matches found by the filter. For this example, every Tag is associated with a single rule. Refer to the moderation Setup & Configuration page for details.

Application name: MyGame. Component Chat:



Configuration for Component Chat

[✓] Store Content?
[ ] User Actions Enabled?

Content Type: [✓ Transient] Persistent

Filtering and Alerting Rules

Harm-Abuse x, PII x, Sexual x, Spam x, Threats x, Vulgarity x, Weapons x

		Generate Alert?
Severe	Allow	☐
High	Allow	☐
Medium	Allow	☐
Mild	Allow	☐

Sending the Request

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 API section.

Request URL

http://localhost:8001/content/item.js

HTTP Method

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

Request Body

Request Attribute	Required?	Description
contentItem.content=<URL Encoded content>	Yes	The content to be filtered.
contentItem.type=text	Yes	For filter requests, the value for type must be **text**.
filter.blacklist.enabled=true	Yes	This enables the blacklist filter.
filter.operation=replace	Yes	The operation used should always be **replace**.
contentItem.application=MyGame	Yes	The application name.
contentItem.component=Chat	Yes	The component name.
contentItem.createInstant=1371683911000	Yes	The unix timestamp in **milliseconds**.

NOTE: Review the API POST Content Item page for details and options for each request parameter.

Handling the 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/or 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.
500	There was a CleanSpeak internal error. A stack trace is provided and logged in the Apache Tomcat /logs/catalina.out file.

Sample cURL Request #1

Content: This message contains bitch and butt

```
curl -X POST -d
"contentItem.content=This+message+contains+bitch+and+butt&contentItem.type
=text&filter.blacklist.enabled=true&filter.operation=replace&contentItem.a
pplication=MyGame&contentItem.component=Chat&contentItem.createInstant=137
1683911000" http://localhost:8001/content/item.js
```

JSON Response #1

```
{
  "contentAction" : "reject",
  "filter": {
    "matched": true,
      "matches": [
        { "start": 22, "length": 5, "type": "blacklist", "locale": "en",
"matched": "bitch", "root": "bitch", "tags":
["Bullying","Sexual","Vulgarity"], "severity": "high", "quality": 1.00000
},
        { "start": 32, "length": 4, "type": "blacklist", "locale": "en",
"matched": "butt", "root": "butt", "tags": ["Sexual"], "severity":
"medium", "quality": 1.00000 }
      ],
      "replacement": "This message contains ***** and ****"
  },
  "moderation": {
    "type": "unmoderated",
    "queued": false,
    "stored": true
  }
}
```

The JSON keys and values are detailed in the Response Reference section.

Sample cURL Request #2

Content: This message contains butt

```
curl -X POST -d
"contentItem.content=This+message+contains+butt&contentItem.type=text&filt
er.blacklist.enabled=true&filter.operation=replace&contentItem.application
=MyGame&contentItem.component=Chat&contentItem.createInstant=1371683911000"
http://localhost:8001/content/item.js
```

JSON Response #2

```
{
  "contentAction" : "authorOnly",
  "filter": {
    "matched": true,
      "matches": [
        { "start": 22, "length": 4, "type": "blacklist", "locale": "en",
"matched": "butt", "root": "butt", "tags": ["Sexual"], "severity":
"medium", "quality": 1.00000 }
      ],
      "replacement": "This message contains ****"
  },
  "moderation": {
    "type": "unmoderated",
    "queued": false,
    "stored": true
  }
}
```

The JSON keys and values are detailed in the Response Reference section.

Result

Notice that the **contentAction** is different for each response. Since the highest severity match in the first message = *high*, the contentAction is *rej ect*. In the second example, the highest severity response = medium resulting in a contentAction of *authorOnly*.

You can choose to ignore every other piece of information returned in the response body and only handle the **contentAction** attribute. If/when a moderation manager modifies the filtering rules, no changes will need to be made to the integration code.

There are five possible contentActions: **reject**, **replace**, **authorOnly**, **queuedForApproval**, and **allow**. Consult with the moderation manager to define what the automated reaction should be in the client application for each possible contentAction.

Images & Video

Coming soon...

Username Filtering & Moderation

Coming soon...

User Registration

Coming soon...

Taking Actions on Users

Coming soon...

System Users

Coming soon...

Concepts & FAQ

Filtering Text

You can control what the filter looks for by setting parameters in the request that is sent to CleanSpeak. CleanSpeak will return a response object (JSON or XML) with detailed information from any match(es) found. Therefore, you can control the ***filter reaction*** by adjusting the filter **request**, parsing the filter **response**, or **both**.

What you can do in the filter request:

- Activate individual filters. The filter types include: blacklist, emails, phone numbers, url addresses, characters, and words.
- Specify what type of blacklist entries to look for such as vulgarities, sexual behaviors, racist remarks, etc.
- Adjust the behavior of the email, URL, and phone number filters.
- And more...

Information that CleanSpeak returns:

- Whether or not a match was found.
- Details about each match that is found.
- The original string with any matches replaced by asterisks.

It is important to note that the filter reaction may differ between content sources. There may also be multiple types of reactions from a single content source after parsing the response.

As you progress to the Examples section, review the Filtering Concepts page of the user guide or consult with the project manager to determine the correct filter reaction(s).

Moderation - Content

Storing & prioritizing content for moderator review requires configuration within the management interface and appending the configuration attributes to the HTTP request that is sent to CleanSpeak. At minimum, a single search engine URL must be stored in System -> Servers and a single Application and single Component must be configured to store content from System -> Applications. The Application name, Component name, and a Unix Timestamp representing the time (in milliseconds) that the content was created are required in the HTTP request.

The moderation request attributes will most often be appended to the same request that is sent to CleanSpeak for filtering content. In some cases, you will send separate filtering and moderation requests. For example, you may chose to queue content differently for the various severity levels of filter matches in the initial filter request and response.

Moderators may be able to action content within CleanSpeak that will have an effect in the client application. Those actions include editing or deleting persistent content such as forum posts or approving/rejecting content such as images. When a moderator actions content, CleanSpeak will send an HTTP POST request to the client. See the Moderation Notifications section for details.

As with filtering, each content source may have unique rules regarding work flow and moderation. The scenarios represented in the Examples sec tion can all be integrated with a single CleanSpeak instance. Review the Moderation Concepts page of the user guide or consult with the project manager to determine the correct moderation work flow for each content source.

Moderation - Users

User details may be stored within CleanSpeak for ease of moderation and to generate reports based on user metrics. New users should be added during the registration process from your application. Each content item has a loosely coupled association with users when the sender UID and/or receiver UID are specified in the request.

Actions may be taken on users from moderators from the user details page in the CleanSpeak management interface. Available actions are configurable from System -> User Actions. When a moderator takes an action on a user, CleanSpeak sends an HTTP POST request to the client as described in the Moderation - Content section above.

Visit the User Registration and Taking Actions on Users sample integration pages for details.

System Users

System users (moderators) may be added to CleanSpeak via the system user API. Visit the System Users page in the sample integrations section for details.

GET Whitelist

The Whitelist service provides a means for retrieving Cleanspeak's whitelisted (allowed) words. This can be very useful for clients that wish to perform character-by-character feedback to their users. For example if a system was using Cleanspeak's whitelist feature they could pull down the whitelist allowed words and as a user types each letter display a positive feedback indicating the word typed so far is acceptable, but as soon as a string is found that is not allowed indicate to the user via a red highlight or other feeback mechanism that the message they are typing will be rejected; once a user has typed in a message consisting of only allowed words, the system should submit that to Cleanspeak to make sure the user hasn't typed in a disallowed phrase.

Please note the response will be large, at the time of this writing there are > 10,000 allowed words, and > 2,000 disallowed phrases; ~1MB.

Request URLs

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

URL	Response Type
http://localhost:8001/filter/whitelist.js	JSON response
http://localhost:8001/filter/whitelist	JSON 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/or malformed. The response will contain a JSON message with the specific errors.
401	WebService authentication is enabled and you did not supply a valid Authentication header. The response will be empty.
500	There was a CleanSpeak internal error. A stack trace is provided and logged in the log file.

Sample Request

URL: http://localhost:8001/filter/whitelist/index.js

To retrieve user information from the GET request:

GET cURL request:

```
curl http://localhost:8001/filter/whitelist/index.js
```

JSON Response:

```
{
  "whitelist":{
    "allowedEntries":[
      {
        "inflect":false,
        "locale":"en",
        "text":"a"
      },
      {
        "inflect":true,
        "locale":"en",
        "text":"aback"
      },
      {
        "inflect":true,
        "locale":"en",
        "text":"abandon"
      },
      {
        "inflect":true,
        "locale":"en",
        "text":"abbey"
      },
      {
        "inflect":true,
        "locale":"en",
        "text":"abdominal"
      },
      {
        "inflect":true,
        "locale":"en",
        "text":"abide"
      },
      {
        "inflect":true,
        "locale":"en",
        "text":"ability"
      },
      {
        "inflect":false,
        "locale":"en",
        "text":"able"
      },
.....  SNIP .....
      }
        "inflect":true,
        "locale":"en",
        "text":"zucchini"
      }
    ],
    "disallowedEntries":[
      {
        "parts":[
          "am",
          "baked"
        ],
        "text":"am baked"
      },
      {
        "parts":[
          "am",
          "he",
          "baked"
        ],
        "text":"am he baked"
      },
      {
        "parts":[
          "am",
          "he",
          "high"
        ],
        "text":"am he high"
      },
      {
        "parts":[
          "am",
          "high"
        ],
        "text":"am high"
      },
........ SNIP .......
      {
        "parts":[
          "you",
          "were",
          "high"
        ],
        "text":"you were high"
      }
    ]
```

```
    }
  }
```

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:

All notifications are sent from CleanSpeak to the notification servers you configure via the Management Interface. These notifications are sent via an HTTP POST using x-www-form-urlencoded data. This is the same format that the CleanSpeak WebService uses to accept requests from your application.

Content 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: `type=moderation&`	String
moderation.application	This parameter specifies the content Application: `moderation.application=MyApp&`	String
moderation.component	This parameter specifies the content Component: `moderation.component=MyComponent&`	String
moderator	This parameter specifies the moderator username that approved/rejected the content: `moderation=admin@inversoft.com&`	

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:
	`moderation.item[0].uid=12345&`
	`moderation.item[1].uid=67890&`
		String

moderation.item[index].action	This parameter specifies the action that was taken on the content. The possible values are **approved** and **rejected**. 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:
	`moderation.item[0].action=approved&`
	`moderation.item[1].action=rejected&`
		String

Example Post Body

```
type=moderation&
moderator=admin%40inversoft.com&
moderation.application=Game&
moderation.component=Forum&
moderation.item%5B0%5D.uid=5&
moderation.item%5B0%5D.action=rejected&
moderation.item%5B1%5D.uid=6&
moderation.item%5B1%5D.action=approved
```

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: `type=userAction&`	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: `userAction.userUID=JoeBlow&`	String
moderator	This parameter specifies the moderator username that took the action on the user: `moderation=admin@inversoft.com&`	String
userAction.application	This parameter specifies the application that the action is being performed in: `userAction.application=MyApp&`	String

userAction.component	This parameter specifies the component that the action is being performed in: `userAction.component=MyComponent&`	String
userAction.action	This parameter specifies the name of the action that's occurring: `userAction.action=mute&`	String
userAction.key	This parameter specifies the name of the key that's occurring for the given action: `userAction.key=30secs&`	String

Example POST Body

```
type=userAction&
moderator=admin%40inversoft.com&
userAction.userUID=1234-ABCD&
userAction.application=Game&
userAction.component=Chat&
userAction.action=Mute&
userAction.key=30+min
```

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: `type=contentItemEdit&`	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: `contentItem.uid=1&`	String
moderator	This parameter specifies the moderator username that edited the content: `moderation=admin@inversoft.com&`	String
contentItem.application	This parameter specifies the application the content was generated in: `contentItem.application=MyApp&`	String
contentItem.component	This parameter specifies the component that the content was generated in: `contentItem.component=MyComponent&`	String

contentItem.content	*This will not be sent when the edited contentItem is a complex type.* This parameter specifies the content that the moderator added as a result of the edit: `contentItem.content=Edited%20by%20moderator&`	String
contentItem.complex[n].name*	*This will only be sent when the edited contentItem is a complex type.* Only sent when the name for this complex data item has been specified upon creation: `contentItem.complex[0].name=The%Title&`	String
contentItem.complex[n].content*	*This will only be sent when the edited contentItem is a complex type.* This parameter specifies the content for the nth complex content item: `contentItem.complex[0].content=The%20brown%20fox%20****%20jumped&`	String

- When complex content is edited all sub-items, even those that weren't modified, will be passed to the notification server.

Example POST Body

```
type=contentItemEdit&
moderator=admin%40inversoft.com&
contentItem.uid=7&
contentItem.application=Game&
contentItem.component=Forum&
contentItem.content=This+has+been+edited..
```

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: `type=contentItemDelete&`	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: `contentItem.uid=1&`	String
moderator	This parameter specifies the moderator username that deleted the content: `moderation=admin@inversoft.com&`	String
contentItem.application	This parameter specifies the application the content was generated in: `contentItem.application=MyApp&`	String

contentItem.component	This parameter specifies the component that the content was generated in: `contentItem.component=MyComponent&`	String

Example POST Body

```
type=contentItemDelete&
moderator=admin%40inversoft.com&
contentItem.uid=8&
contentItem.application=Game&
contentItem.component=Forum
```

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 with a request body containing x-www-form-urlencoded data 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 an HTTP response status code of 200. If there was any type of error or failure, your WebService must send back an 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 are configured via the Management Interface under System -> Notification Servers. If you have multiple notification servers configured for a single Component, they must all correctly handle the notification for the transaction to succeed.

PHP Specifics

The PHP $_POST and $_REQUEST global variables cannot be used in their default form to handle notifications from CleanSpeak due to PHP replacing dots with underscores. Use the following function to fix the $_POST variable prior to processing the data:

```
fix($_POST, file_get_contents('php://input'));

function fix(&$target, $source) {
  if (!$source) return;
  $target = array();
  $source = preg_replace_callback('/(^|(?<=&))[^=[]+/', function($key) {
    return bin2hex(urldecode($key[0]));
    }, $source);
  parse_str($source, $post);
  foreach($post as $key => $val)
    $target[ hex2bin($key) ] = $val;
}
```

Credit: http://stackoverflow.com/questions/68651/get-php-to-stop-replacing-characters-in-get-or-post-arrays

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.

To download or check on latest updates, please login to http://www.inversoft.com/login and view our Downloads page:

http://www.inversoft.com/account/list-downloads

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

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

NOTE: Running the update script will shut down the CleanSpeak service if it is running. The service will need to be restarted after the update is finished.

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 located in the migrations folder inside the Database Schema ZIP file. This file can be downloaded by logging into your account at www.inversoft.com.

Warning
When upgrading your database from a previous version, be sure to only run the scripts located in the `migrations` folder, the base files `mysql.sql` and `postgresql.sql` should only be used during a clean installation when no database schema is present.

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 Version 2 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*.*Post* *greSQL 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 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	**contentItem.content**
*request.type	**contentItem.type**
request.filter.enabled request.filter.cleanspeakdb.enabled	**filter.blacklist.enabled**
request.filter.operation	**filter.operation**
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 Version 2*
***Categories (and Contexts) have been changed to Tags.*

Upgrading from 2.x

If you are upgrading an installation of CleanSpeak from 2.0, 2.1, or 2.2, you will need to reconfigure CleanSpeak.

Configuration

We moved all of the configuration for CleanSpeak out of server.xml and into a new configuration file. This new configuration file is shared by all CleanSpeak services that run on the same server. On Linux, the location of the new configuration file is:

```
/usr/local/inversoft/config/cleanspeak.properties
```

This file is automatically created when you install or upgrade CleanSpeak to 2.3. All you will need to do is edit this file in your favorite text editor (vim, emacs, pico, etc.) and change the settings to match your installation. You can find detailed information about each configuration property inside the file or in the Configuration page of the documentation.

License File & ID

The license file system has been updated to include a call-back to Inversoft to retrieve license information. The primary reason for making this change is to make installation and configuration easier for you. When your account is renewed or updated, your license will automatically be updated for you.

If you'd prefer, you can still manage your own license file similar to the old process as well.

Details can be reviewed: License ID

Filter Logic Updates

There are three new configuration options for blacklist entries: Ignores, Replace Phonetics?, and Collapse Doubles?. Also, the whitelist has been renamed to the Blacklist Dictionary. Note that there is still a Whitelist link, but this is a different list entirely and used for the new whitelist feature.

Ignore cases are now used where the whitelist was used to ignore specific match cases. For example, if you want to filter **cok** but ignore *cook*, then you need to add "cook" as an Ignore case to the **cok** entry. With prior CleanSpeak versions, you could add *cook* to the whitelist to prevent it from being filtered. This is no longer the case! Review the Add/Edit Blacklist Entries page for details.

After migrating your blacklist to 2.3 and updating CleanSpeak, you may need to add ignore cases to existing entries. You may also chose to install our latest default filtering lists which have been improved from prior versions.

Contact support@inversoft.com for details.

Reindexing

IMPORTANT NOTICE

If you are storing content with the moderation platform, you must run the reindexer from the CleanSpeak Management Interface. This is located under System -> Search Index. Contact support@inversoft.com for details and recommendations.

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 inversoft/logs directory. In most cases all errors will be written out to the cleanspeak-<service>.log file. You can email support@inversoft.com to help troubleshoot any possible errors that might exist.

When contacting Inversoft support, zip the contents of the logs directory and attach it to the email.

User Guide

Welcome to the CleanSpeak 2.3 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 list management.

For technical information, refer to the Technical Guide.

Next Step: Concepts

Search this guide:

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 s ection 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 list of disallowed 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 disallowed phrase 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 also checking for disallowed 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.

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 (Pre-Moderation)

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

Alert Queue (Post-Moderation)

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 Tag(s) and various severity levels 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).

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 queue 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.

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
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.

Also, each user needs to have a unique account when using the moderation system. Proceed to System Users & Permissions to add accounts.

The management interface has four 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

Reports

- View and compare individual moderator activity
- View top offenders and top content producers
- Export reports to a spreadsheet

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 Moderator and Moderation Manager roles, the moderator can be restricted to particular application(s). If no applications are selected, then the moderator will have access to all applications.

Important Note: Each moderator needs to have a unique account when using the moderation system.

Filter Roles

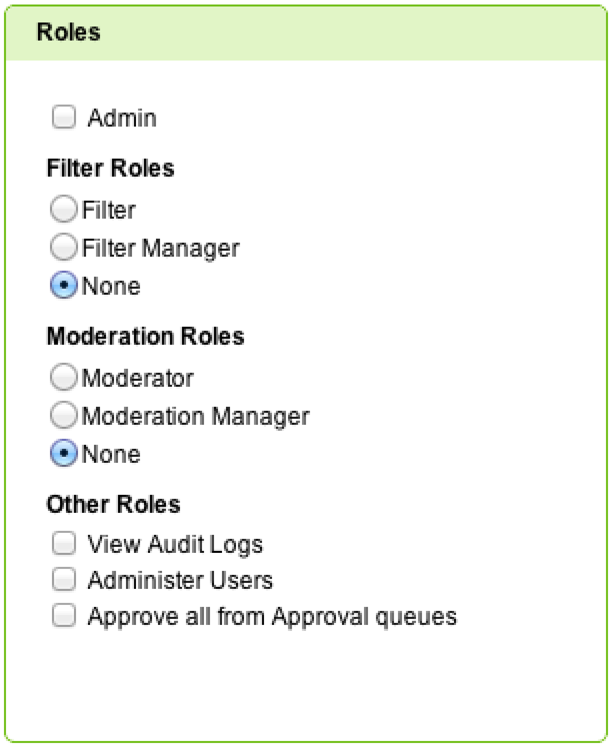
- **Filter:** Modify the black and whitelists. Use the verification tool to test the filter. Cannot approve list changes.
- **Filter Manger:** All of “Filter” role abilities. Can also approve changes made to the filter lists, edit tags, and import/export lists.

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.*
- **Approve all from Approval queues:** Allow the moderator to Approve or Reject all displayed content within approval queues rather than moderating content individually.


Roles

☐ Admin

Filter Roles

○ Filter
○ Filter Manager
● None

Moderation Roles

○ Moderator
○ Moderation Manager
● None

Other Roles

☐ View Audit Logs
☐ Administer Users
☐ Approve all from Approval queues

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 severity levels and tags to your business requirements. You can jump to detailed list management pages with the links below.

- Blacklist Search
- Add/Edit Blacklist Entries
- Filter Mode
- Testing the Filter
- Approving Changes
- Dictionary
- Import/Export Lists
- Multilingual
- Whitelist

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. When defining your filtering actions, you can take different actions based on Tags. Tags can be added, edited, or deleted via *Filter -> Tags*. Also, the moderation system generates alerts from Tags as described in the Moderation Concepts sec tion. The following table lists the Tags that are included with the blacklist that is initially installed:

Tag Name	Description
Account	Words and phrases used when attempting to share account information, such as *password* and *username*.
Alcohol-Drug	Words and phrases with alcohol and drug connotations.
AsciiArt	Combinations of symbols that are used to convey sexual and other inappropriate images.
Bigotry-Racism	Words and phrases with racist, religious, and other bigotry type connotations.
Bullying	Words and phrases used when a user attempts to bully another user.
Grooming	Words and phrases used by sexual predators to groom another user.
Harm-Abuse	Words and phrases used when a user communicates about harming themselves or another.
PII	Words and phrases used when attempting to share personally identifiable information (PII).
Sexual	Words and phrases with sexual connotations.
Spam	Words and phrases used by spammers.
Threats	Words and phrases used when a user attempts to threaten another user.
Vulgarity	Words and phrases that are profane or inappropriate slang.
Weapons	Words and phrases used to communicate about weapons.

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: Blacklist Search

Blacklist Search

Blacklist Search Form

The blacklist search form is accessible via the Management Interface in the left column of *Filter -> Blacklist*. The following is a description of each field:

Search Field	Description
Text	Full or partial text searches for blacklist entries and variations of blacklist entries.
Locale	The two digit language code and optional two digit country code. Examples: **en**, **en_US**, **es**, **es_MX**
Severity	Find only entries with a specific severity.
Tags	Find only entries with specific tags. Note: When adding blacklist entries, at least one tag must be assigned to the entry. However, there are cases when entries may not contain any tags. Checking **None** will uncheck all other tags to be able to search for entries that do not have any tags.
Parts of Speech	Find only entries with specific parts of speech.
Filter Mode	Find only entries with a specific filter mode.

Clear Button	Clears the search form and displays all blacklist entries in the main display.

Deleting Multiple Entries

As mentioned in the CleanSpeak 101 page, the blacklist that is initially installed contains many entries that are specific to filtering children's applications. You may want to bulk delete specific tags and/or severity levels for applications targeted to mature audiences or if you have other unique filtering requirements.

To bulk delete multiple blacklist entries:

1. Determine the type of entries you want to delete. Refer to the List Management page for details of severity levels and tags.
2. Display all entries from this criteria with the search form described above.
3. Note the amount of results being displayed. Show 100 at a time if needed.
4. *In the results pane, check the box just below the **Add Entry** button to select all entries.
5. Click **Delete Selected** to remove all selected entries.
6. Approve the changes as described in the Approving Changes section.

***NOTE**: Many entries have multiple tags. The blacklist entry "ass", for example, has two tags: Vulgarity and Sexual. If you intend to delete all entries from "Sexual" by performing a blacklist search to find all entries with the tag "Sexual", the results will include "ass". Prior to deleting all selected content in step 5 above, scroll through the list and uncheck any entries that have more than one tag if desired.

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.

[Basics] [Tags] [Miscellaneous]

Basics

Text (Required): Base Entry.

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.

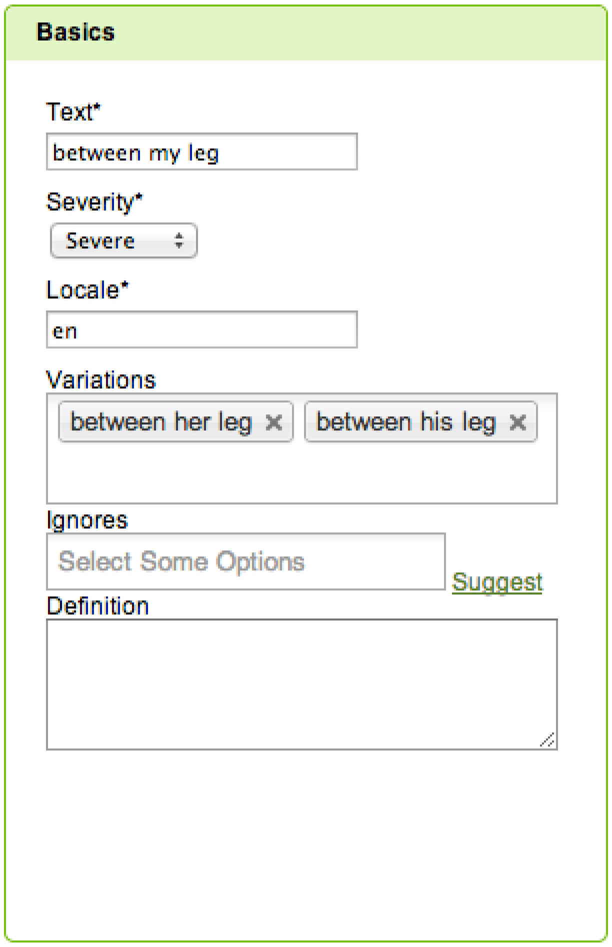
Examples: **en** = English, **es** = Spanish, **en_US** = American English, **es_MX** = Mexican Spanish.

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. Each variation must be unique. Note that the filter will look for inflections based on the selected parts of speech for variations as well as the base entry.

Ignores (Optional): Field used to add cases that the filter should ignore when it finds a match on this entry. Clicking the *Suggest* link will display a list of suggested ignore cases.

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


Basics

Text*

between my leg

Severity*

Severe

Locale*

en

Variations

between her leg ✕ between his leg ✕

Ignores

Select Some Options

Suggest

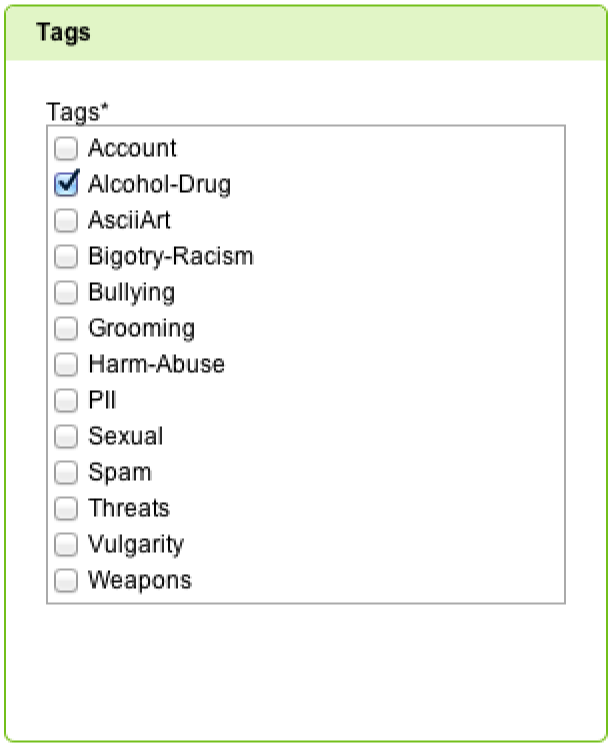
Definition

Tags

Categories of entries. An entry may have multiple tags but must have at least one. Tags may 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*

- [ ] Account
- [x] Alcohol-Drug
- [ ] AsciiArt
- [ ] Bigotry-Racism
- [ ] Bullying
- [ ] Grooming
- [ ] Harm-Abuse
- [ ] PII
- [ ] Sexual
- [ ] Spam
- [ ] Threats
- [ ] Vulgarity
- [ ] Weapons

Miscellaneous

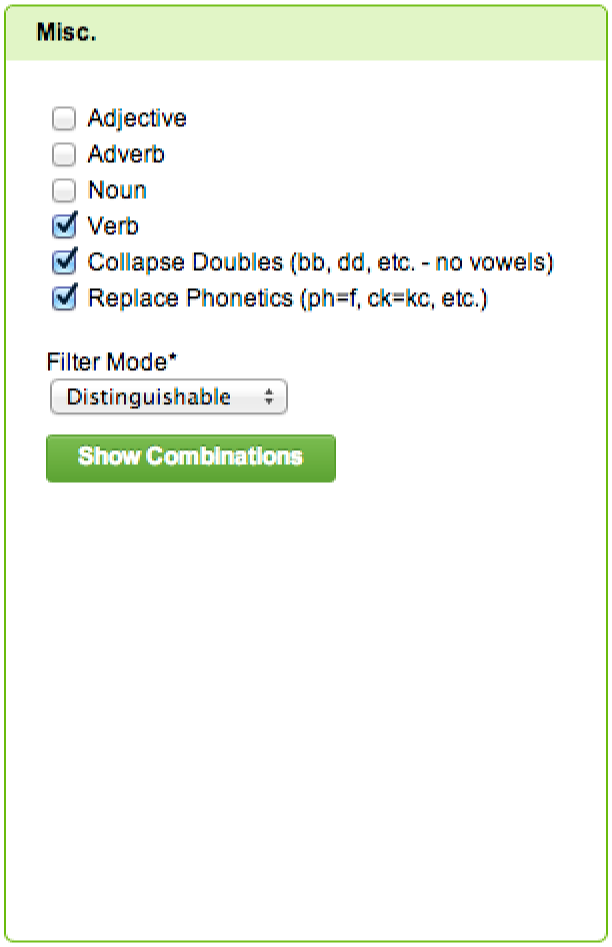
Parts of Speech: Used to find inflections of root entries. Check the part(s) of speech if you want CleanSpeak to find inflections for you automatically based on the locale. Currently, CleanSpeak provides inflection support for English, Spanish, French, German, Italian, Dutch, Polish, Russian, & Swedish.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

Collapse Doubles: This tells the filter to look for cases of consonants in a singular version when they are repeated in an entry. For example, if CleanSpeak should look for *bal* from the blacklist entry *ball*, check Collapse Doubles. In most cases, this should be checked.

Replace Phonetics: This tells the filter to look for phonetic replacements automatically. For example, if the entry has an "f", the filter will automatically look for "ph" in place of the "f" if this is checked.

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

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
- [x] Verb
- [x] Collapse Doubles (bb, dd, etc. - no vowels)
- [x] Replace Phonetics (ph=f, ck=kc, etc.)

Filter Mode*: Distinguishable

Next Step: Filter Mode

Filter Mode

Use Case

The goal of a text filter is to find words that are undesired but not to filter words that shouldn't be filtered. The table to the right illustrates examples of matches that should be found and those that should **not** be found (referred to as *false-positives*) with the blacklist entries ***gin***, ***ass***, and ***fuck***.

CleanSpeak understands how to filter words in these situations based on the *Filter Mode* that is assigned to each entry. Below is a list of how you might describe the situation where you want each word to be filtered:

- **ass**: (Embeddable) This word should be filtered when it is by itself, when replacement characters are used, and when it's used to create a new bad word by adding numbers or words to the front or back without spaces.
- **fuck**: (Distinguishable) This word should be filtered no matter what! It doesn't matter if it is surrounded by letters or words.
- **gin**: (Not-Embeddable) This word should only be filtered when it is by itself (spaces on both sides).

Original Message	Should there be a filter match?
You are an **ass**	Yes
Don't be an **a$$**face	Yes
I **ass**oom that's right	No
idontgiva**f|_|ccck**doooo d	Yes
go**f uc k**off	Yes
Do you like **gin**?	Yes

Hangggg **in** there!	No
That's a bar**gin**!	No

Filter Modes Explained

	Exact Match	Not-Embeddable	Embeddable	Distinguishable
Definition	Only find this entry when it is literally an exact match with spaces on either side.	Find this entry with spaces on either side and also look for repeat characters, replacement characters/symbols, and separators	Find this entry when it is embedded next to another word (without spaces), when numbers are embedded before or after, or when repeat characters, replacement characters/symbols, and separators are used.	Find this entry in all cases: Whether it's embedded next to words, numbers, random characters, and/or special characters, repeat characters/symbols, or separators are used.
Sample Entry	**f'er**	**gin**	**ass**	**fuck**
Do Filter	• f'er	• gin • giiinnnn • g!!n	• assface • bigAss123 • a$$face	• fuuccckkkk • foobar231FuCkblah
Do NOT Filter	• fer • f'er3 • f'3r	• gin123 • bargin	• assoom • bassguitar	

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*.

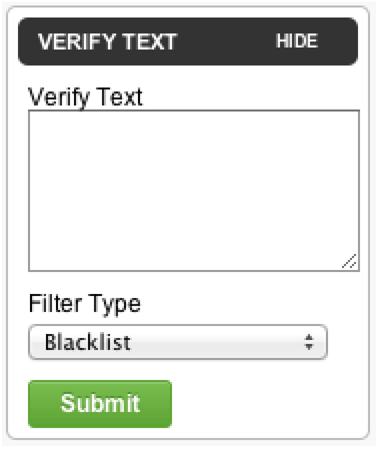
[Verify Text Box] [Match Results] [Troubleshooting False-Positives] [Other Possibilites] [WebService Test Form]

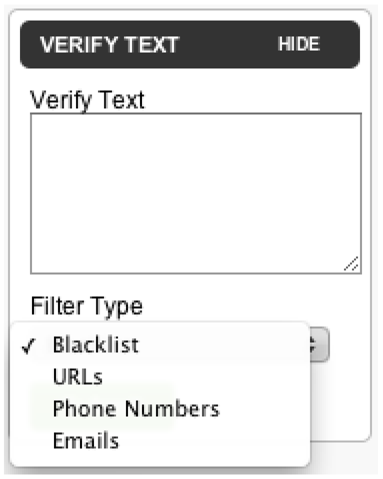
Verify Text Box

The Verify Text box is accessible from every page under *Filter*. Add content to the text box by typing or copy/pasting from another source. Multiple lines of content may be tested at a time.

You may select which filter type you would like to filter against by using the Filter Type select box. Only one of the following filters can be tested at a time: Blacklist, URLs, Phone Numbers, and Emails.

If a match is found, the result(s) will be listed in the Match Results page as displayed below.





Match Results

When a match is found, it is highlighted with a red background in the result pane. Mousing over the match will display information on the match. If the Filter Type is Blacklist, 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 use the WebService Test Form described below.
   a. If the change is not taking effect, review Other Possibilites below.

Other Possibilites

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. Click *Filter*.
5. Results will be displayed under *The result*. Any matches found by the filter will be replaced with asterisks.

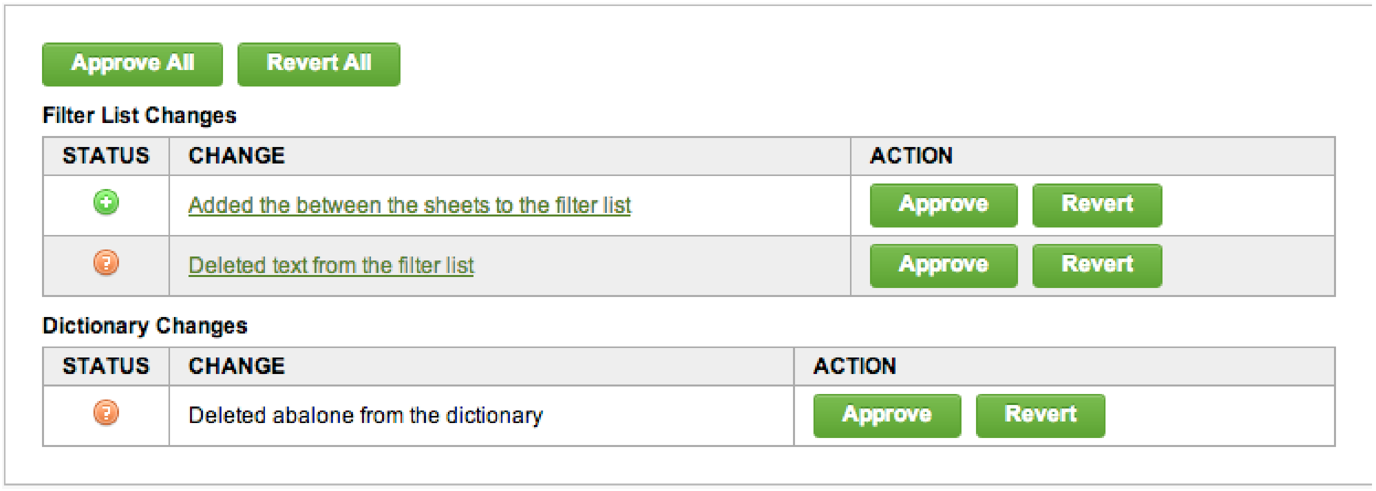
*Enhancements to this form will be coming soon!

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.

Clicking Filter List changes will take you to the add/edit entry screen to review the configuration of the entry as needed.



Filter List Changes

STATUS	CHANGE	ACTION
⊕	Added the between the sheets to the filter list	Approve Revert
?	Deleted text from the filter list	Approve Revert

Dictionary Changes

STATUS	CHANGE	ACTION
?	Deleted abalone from the dictionary	Approve Revert

Changes may also be reverted which allows flexibility to experiment with the lists without the fear of breaking the database. Approving or reverting changes may be performed per entry or you can approve or revert all with the Approve All and Revert All buttons.

Next Step: Dictionary

Dictionary

CleanSpeak utilizes the blacklist dictionary for embedding cases and other advanced filtering logic. Updates to the dictionary should not be necessary except for rare occasions. For further details, contact support@inversoft.com.

Note: The dictionary is not used as an ignore set like the *Whitelist* was in previous versions of CleanSpeak. Instead, each blacklist entry can have Ignore cases associated to them. See Add/Edit Blacklist Entries for details.

Next Step: Import/Export Lists

Import/Export Lists

Importing Lists

A properly formatted JSON file may be imported via *Filter -> Import*. Once the import is complete, all new entries will be in a pending approval state and need to be approved via Filter -> Approvals.

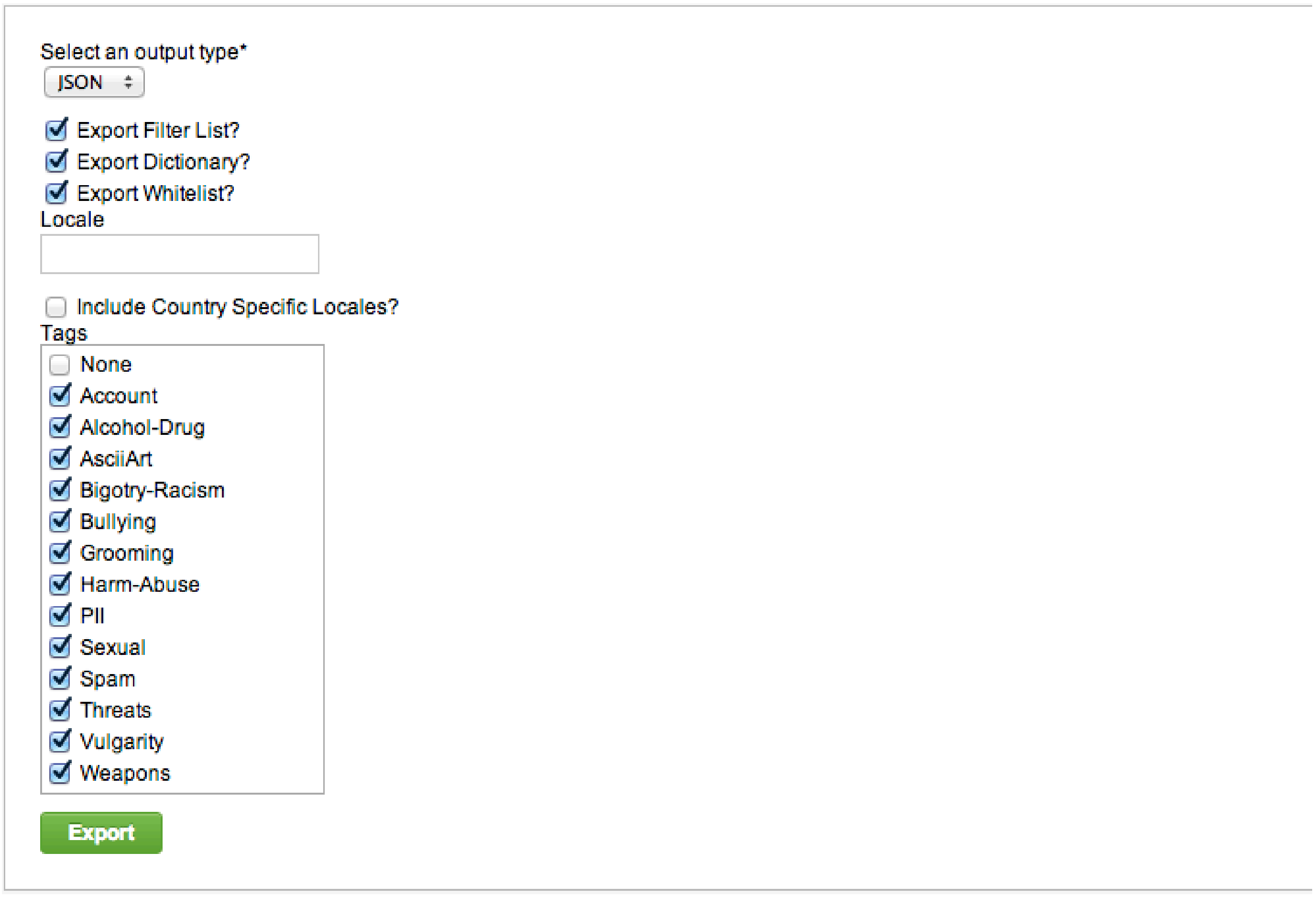
Importing a List with Duplicate Entries

You may already have filtering lists installed and chose to import another list. This would be most common when Inversoft releases updates to the default lists that are initially installed.

During the import process, CleanSpeak will check for duplicate entries from the new list against the existing list. When a duplicate entry exists, CleanSpeak will overwrite the existing entry with the new entry only when the entries have different attributes, including Severity, Variations, Tags, Parts of Speech, and Filter Mode. You may choose to keep the original entry by reverting the change during the approval process.

Exporting Lists

You may also export your list into an JSON, SQL, or CSV file via *Filter -> Export*. Your entire black and white lists may be exported or you can specify a locale and/or entries that have specific tags.



Select an output type*
JSON

- [✓] Export Filter List?
- [✓] Export Dictionary?
- [✓] Export Whitelist?

Locale

- [ ] Include Country Specific Locales?

Tags

- [ ] None
- [✓] Account
- [✓] Alcohol-Drug
- [✓] AsciiArt
- [✓] Bigotry-Racism
- [✓] Bullying
- [✓] Grooming
- [✓] Harm-Abuse
- [✓] PII
- [✓] Sexual
- [✓] Spam
- [✓] Threats
- [✓] Vulgarity
- [✓] Weapons

Opening CSV Exports with MS Excel

Exporting to CSV will generate a UTF-8 Encoded .csv file. In order to open the file with Excel, the Import Wizard must be used to set the file encoding to UTF-8. Contact us for further assistance.

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, Portuguese, Polish, Swedish, Danish, Norwegian, & Finnish. We also have blacklists for Arabic and Japanese. Since these languages do not have regular forms to inflect, no inflection support is required (See the FAQ below). 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 as an Ignore. 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

camina	caminarais	caminaríais	camines
caminaba	caminaran	caminaríamos	camino
caminabais	caminaras	caminarían	caminá
caminaban	caminare	caminarías	caminábamos
caminabas	caminareis	caminas	camináis
caminad	caminaremos	caminase	camináramos
caminada	caminaren	caminaseis	camináremos
caminadas	caminares	caminasen	caminás
caminado	caminaron	caminases	caminásemos
caminados	caminará	caminaste	caminé
caminamos	caminarán	caminasteis	caminéis
caminan	caminarás	caminastes	caminés
caminando	caminaré	camine	caminó
caminar	caminaréis	caminemos	
caminara	caminaría	caminen	

Multilingual FAQ

How do I install additional languages that you support?

- Languages are imported with the management interface from ***Filter -> Import***. The language files are available for download from your Downloads Page under **Database Downloads**. Click the *Languages Database* link to download a zip file that contains each language we support (other than English- that gets installed when CleanSpeak is initially set up).

Should I install all of your available languages?

- ***You certainly can, but we only recommend adding languages that are relevant to filtering for your community.***

How can I specify what language(s) to filter against?

- If you are using the CleanSpeak moderation platform, you can specify the language(s) to filter against *per application* by configuring Filter Rules.
    - For most use cases, you will want to create separate applications for each language. For example, if you're integrating a forum, you most likely have your forums separated by locale. For each locale, create a separate CleanSpeak Application such as "Forum English", "Forum Spanish", etc. If you don't separate your online community by locale, see the next question below.
- If you are using the CleanSpeak filter only, CleanSpeak will respond with every match from every language that's installed in your blacklist. To specify languages to filter against, parse the response object and check the locale of each match. Only action matches that fulfill your criteria.
    - Note- if you'd like to replace matches for x language(s) only with asterisks (or another character), you'll need to build the string manually by using the start and length of each match that has x locale. For further assistance, contact support@inversoft.com.

Can a blacklist entry be configured differently for different languages?

- Yes. Each blacklist entry is identified in CleanSpeak by the text AND locale (primary key). For words that have different severities, tags, filter mode, etc for different languages, add the same word multiple times but configured differently in your blacklist.
    - **IMPORTANT**: Be careful of the inflections that CleanSpeak attempts to generate based on the locale + part of speech of an entry. For example, if you add **smurf** to the English blacklist and mark it as a *noun* and *verb*, then CleanSpeak will know to look for ***smurfs***, ***smurfed***, ***smurfing***, etc. However, if you add smurf as a **Spanish** noun and verb, CleanSpeak will **not** find those inflections because smurf would not be inflected that way according to the Spanish language. If you want to filter smurf and smurfed in Spanish, then you'd need to add smurfed as a variation to smurf (and the locale would be set to **es**).

I added a new entry but I can't seem to find it with the blacklist search form. What's the deal?

- Many multilingual entries have special characters such as *cabrón* with an ó in place of an o. When using the blacklist search form, you must search for the entry *with* the special characters. In this example, searching for cabron will not display the cabrón entry in the results- you'd need to search for the text *cabrón*.
- Alternatively, you could use the **Verify Text** box to perform a filter test on the word *cabron*. Unless the cabrón entry has ***cabron*** included as an **Ignore**, CleanSpeak will find cabrón as a filter result from cabron.

Do you have language detection?

- No. The reason why language detection doesn't work very well with text filtering is that users can (and will) mix languages in a single message to try and trick the filter.
    - For example, imagine a chat message that has 10 words, 9 are English and the last word is a Spanish profanity. A language detection algorithm would (correctly) assume that the language of the message is English. If we only filtered against the English blacklist, then the Spanish profanity would "get through" the filter.
- The good news is that "language collisions" aren't very common. A language collision occurs when a word is *bad* in one language but *ok* in another language. Most of the time, you'll be able to determine if an entry should be filtered all the time because it's unique or simply globally understood as a profanity. If you're using the CleanSpeak moderation platform, you can easily customize filter rules to specify the filtering and moderation workflow for messages that contain different types of blacklisted entries for various languages.

- When language collisions do occur, there are few techniques you can employ. One example is the word ***con***, which is a very normal/common word in Spanish (*con* means "with"), but it's a profanity in French. If you have a community that speaks both Spanish and French, what do you do when *con* is found in a message? For this example, since the word is extremely common in Spanish and would cause too many false-positives in Spanish chat messages if the word is filtered, you'll want to let the word through. However, you could also add common **phrases** to the French blacklist with the word *con* in them to catch it used negatively in French as often as possible. For other word collisions, you may decide to filter the word because it's most often used inappropriately. Lastly, you could also chose to let the word through and generate a user alert instead when the word is used for a moderator to review later.

How do I add entries for a language that you do not officially support?

- CleanSpeak can filter any language that's in the UTF-8 character set (essentially ALL global languages). Using the Blacklist Add/Edit Page as a guide, create a new blacklist entry and set the locale to be the language of the entry you're adding and remember that CleanSpeak will ***not*** inflect the word based on the part of speech. For example, if you want to add to your blacklist, the Nepali word for "breasts", add "" as the text and "ne" as the locale. For any variations of the word whether they are inflections, misspellings, etc, add them individually to the **Variations** field.

What is the process for adding new supported languages to CleanSpeak?

- There are two steps to adding language support to CleanSpeak: Building a blacklist and adding inflection support.
    - Blacklist: We can build a blacklist for any language with our network of partners and moderation experts to suit your requirements. Or, you can build your own blacklist after the inflection support is added (below). If you build your own blacklist, you will have sole ownership of the list and do not need to share it with us if you'd prefer not to!
    - Inflection Support: For languages that have ***regular forms*** of stems that can be inflected, we can add intelligence so that CleanSpeak filters inflections based on the part of speech of the root entry (see the *caminar* example at the top of the page). Languages have varying degrees of difficulty to accomplish this task, so we would need to investigate the language in question before providing an estimate. Many languages such as Japanese, Arabic, and others simply do not have a concept of "inflection", so this step would be skipped when adding the *supported* language to CleanSpeak. Instead, only the blacklist would need to be built.
    - For details, contact sales@inversoft.com.

Whitelist

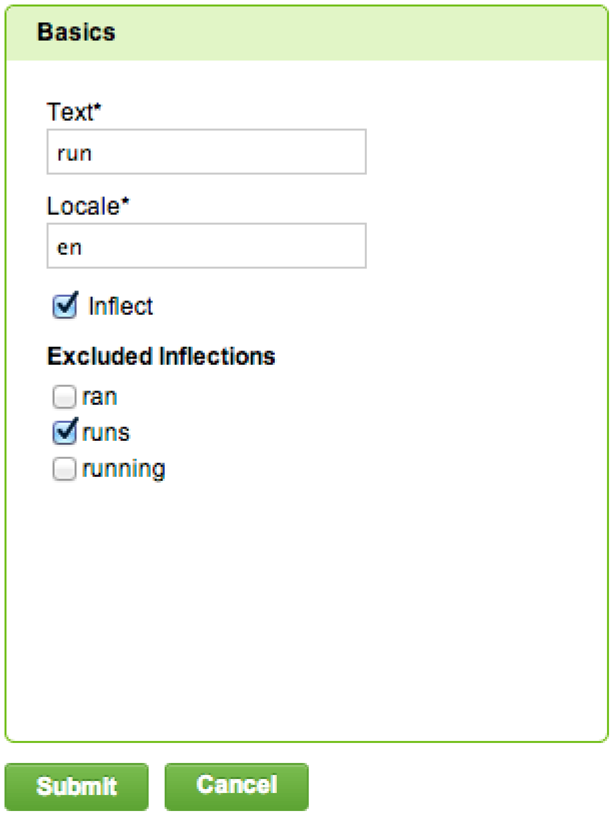
When implementing a whitelist (restrictive) message system, words are only allowed to be used that are on a pre-approved list. There must also be a list of phrases that are ***not*** allowed which are composed of allowed words. For example, the words *in*, *my*, and *hole* may all be on the pre-approved list, but the phrase ***in my hole*** should not be allowed.

You can manage the list of approved words and disallowed phrases from Filter -> Whitelist in the management interface as described below.

For details integrating the whitelist into your application, visit the GET Whitelist and Content Item API pages.

Allowed Words

Allowed words require a locale to generate infections of the word. There is the option to inflect words, and if **Inflect** is checked, there is a selection box to mark individual inflections as Exclusions. For example, if you want to add *run* to the allowed words list, and want its inflections to also be whitelisted, but you didn't want to allow *runs*, you'd want: inflections checked, and *runs* checked in the exclusions list.

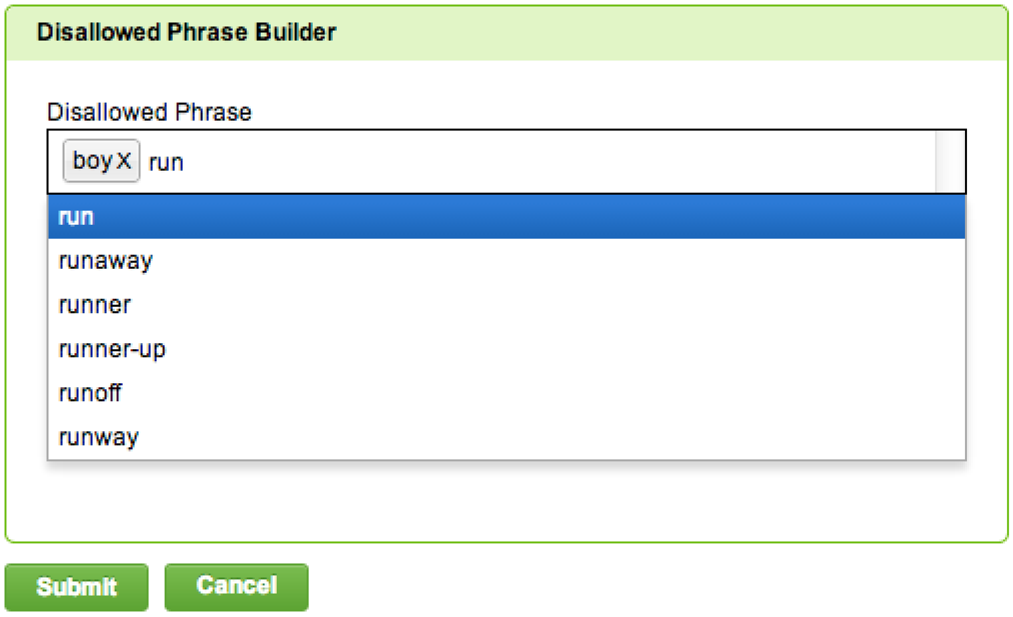


Disallowed Phrases

Disallowed phrases can only consist of words on the Allowed Words list. Each allowed word in a disallowed phrase will be inflected if the allowed word is configured to be inflected. Example: if *boy run* is on the disallowed phrase list and "run" is inflected, "boy runs," "boy running," "boy ran" would all be marked as disallowed phrases.

The phrase builder

As the user types a list of possible allowed word matches will be displayed for the user to create their phrase.



To complete a disallowed phrase click **Submit**.

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 sec tion). Queue content is accessed from the Moderation Dashboard.

Next Step: Setup & Configuration

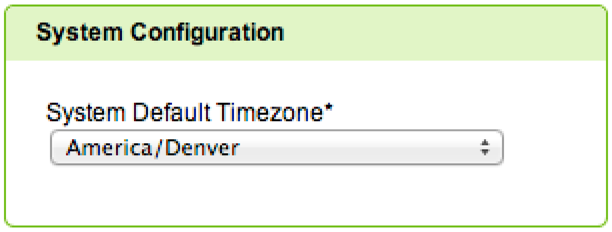
- Setup & Configuration
- Dashboard
- Queues
- Escalations
- Threaded View
- Users & Actions
- Content & User Search
- Edit & Delete

Setup & Configuration

- System Configuration
- Applications & Components
    - Component Settings
    - Filter and Alerting Rules

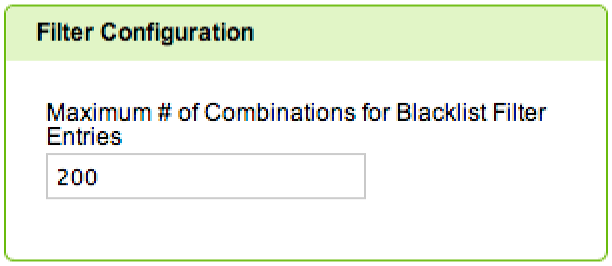
System Configuration

System -> Configuration


System Configuration

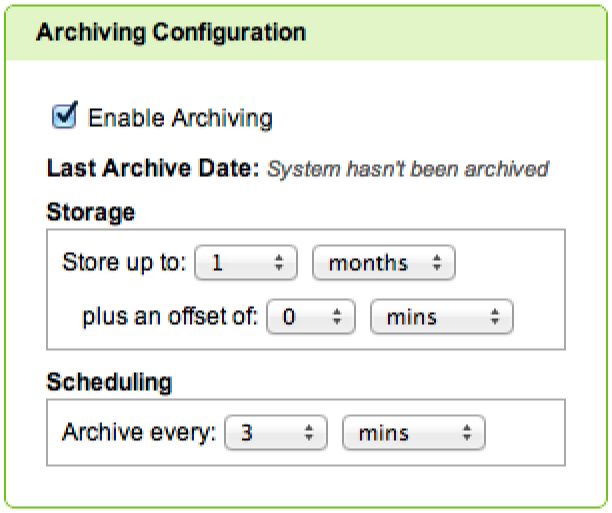
System Default Timezone*
America/Denver

System Default Timezone: Set to the application time zone.


Filter Configuration

Maximum # of Combinations for Blacklist Filter Entries
200

Blacklist Filter Combinations: The number of possible combinations allowed from a single blacklist entry.


Archiving Configuration

☑ Enable Archiving

Last Archive Date: *System hasn't been archived*

Storage

Store up to: 1 months

plus an offset of: 0 mins

Scheduling

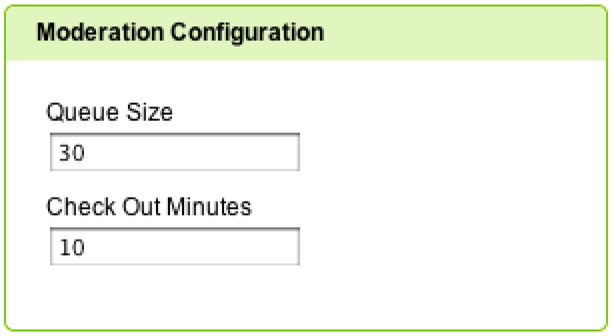
Archive every: 3 mins

Enable Archiving: Checking this will instruct CleanSpeak to delete all content from the database beyond a specified date range.

Last Archive Date: Date and time when the last archive occurred.

Storage: All content prior to the "Store up to:" setting from the current date/time will be deleted. Add an offset for the archival process to run at a different day/time than the day/time of the content itself. For example, set the offset to 12 hours for content that is stored at 5pm to be archived at 5am.

Scheduling: Instructs CleanSpeak how often archiving should occur.


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

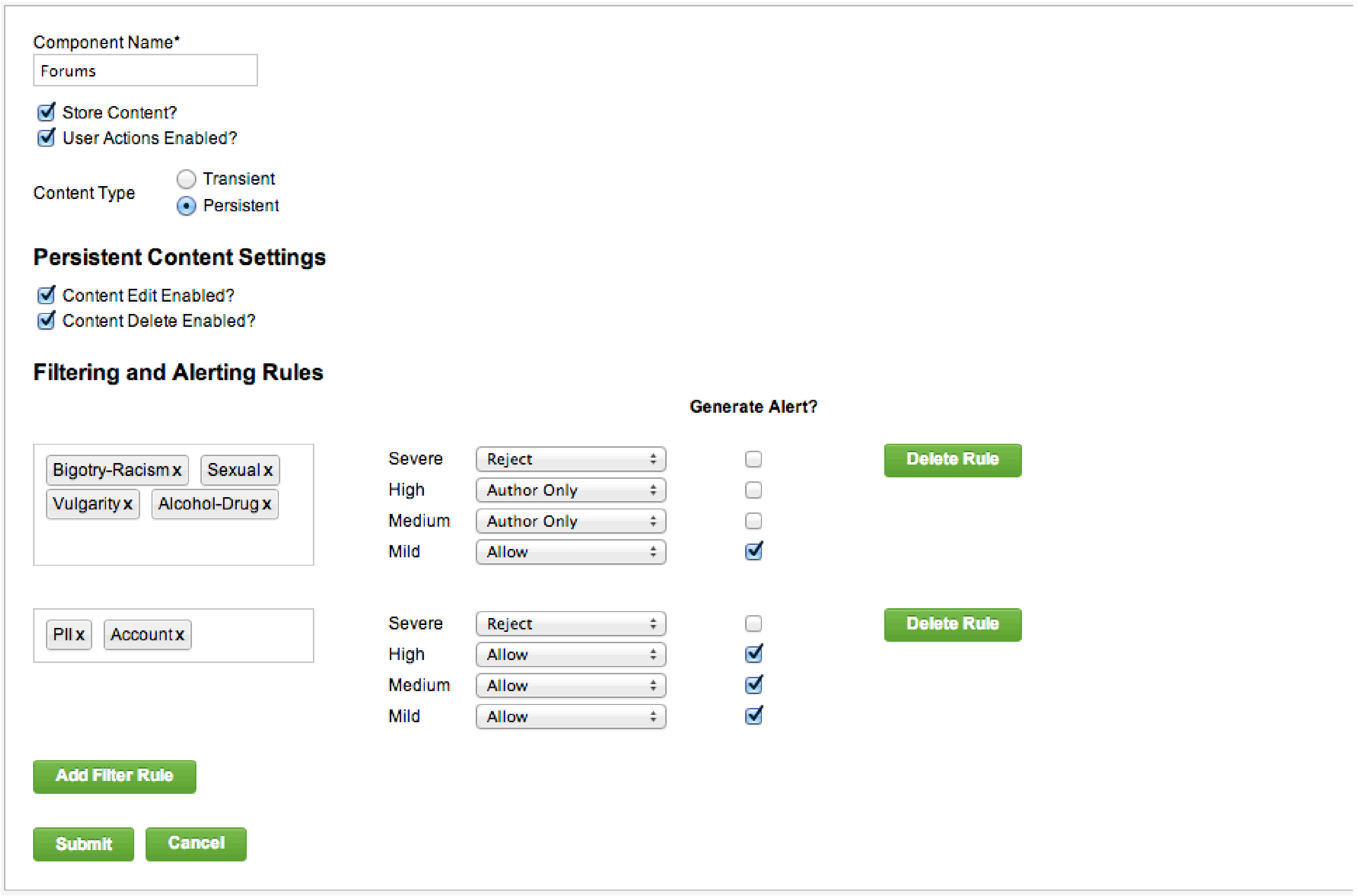
System -> Applications

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 business requirements.

Begin by creating your first Application by clicking **Add Application**. Each Application can have multiple content sources defined as *Components*. For example, you might name your application "MyGame" which will have multiple components: "Chat", "Forums", and "Usernames".

Component Settings

Click **Add Component** to begin. Refer to the table below that describes each configuration option.


Component Name*

Forums

☑ Store Content?
☑ User Actions Enabled?

Content Type ○ Transient [✓ ● Persistent]

Persistent Content Settings

☑ Content Edit Enabled?
☑ Content Delete Enabled?

Filtering and Alerting Rules

Tags	Severity	Reaction	Generate Alert?	
Bigotry-Racism x, Sexual x, Vulgarity x, Alcohol-Drug x	Severe	Reject	☐	Delete Rule
	High	Author Only	☐	
	Medium	Author Only	☐	
	Mild	Allow	☑	
PII x, Account x	Severe	Reject	☐	Delete Rule
	High	Allow	☑	
	Medium	Allow	☑	
	Mild	Allow	☑	

Configuration Option	Description
Component Name	Name of the component.
Store Content?	Check to store incoming content in CleanSpeak. This is required for all content moderation, including editing content, deleting content, queueing content for approval, and generating alerts.
***User Actions Enabled?**	Check to give moderators the ability to take actions on users for this component from the user details page.
Content Type	Transient: Content that will not be available for editing, deletion, or pre-moderation such as real-time chat. Persistent: Content that is published for repeated views, such as forum posts, product reviews, and public display names (usernames).
***Content Edit Enabled?**	If the Content Type is Persistent, this allows for moderators to modify the text. CleanSpeak will store prior version(s) of content that is edited.
***Content Delete Enabled?**	If the Content Type is Persistent, this allows for moderators to delete the text. CleanSpeak will continue to store deleted content for review at a later time.

*Requires a Notification Request Handler.

Filter and Alerting Rules

You can configure what should *automatically* happen to content (in real-time) by setting up Filter Rules. Note that each Component can have unique Filter Rules. Refer to the Filtering Concepts guide for ideas on preparing your filter reactions.

Begin by clicking **Add Filter Rule**. Each rule is defined first by the Filter List Tag(s) associated with the Rule. When clicking in the box to the left, all Tags will be displayed that you can select from. A Tag can be associated with only one Rule, but a Rule can be associated with multiple Tags.

After selecting the Tags for the rule, configure what the filter reaction should be when there is a filter match for each severity level. Next, determine if you want to generate an alert for matches with each severity level.

Add multiple rules by clicking Add Filter Rule again.

The filter rule will be returned in the Filter Response.

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.

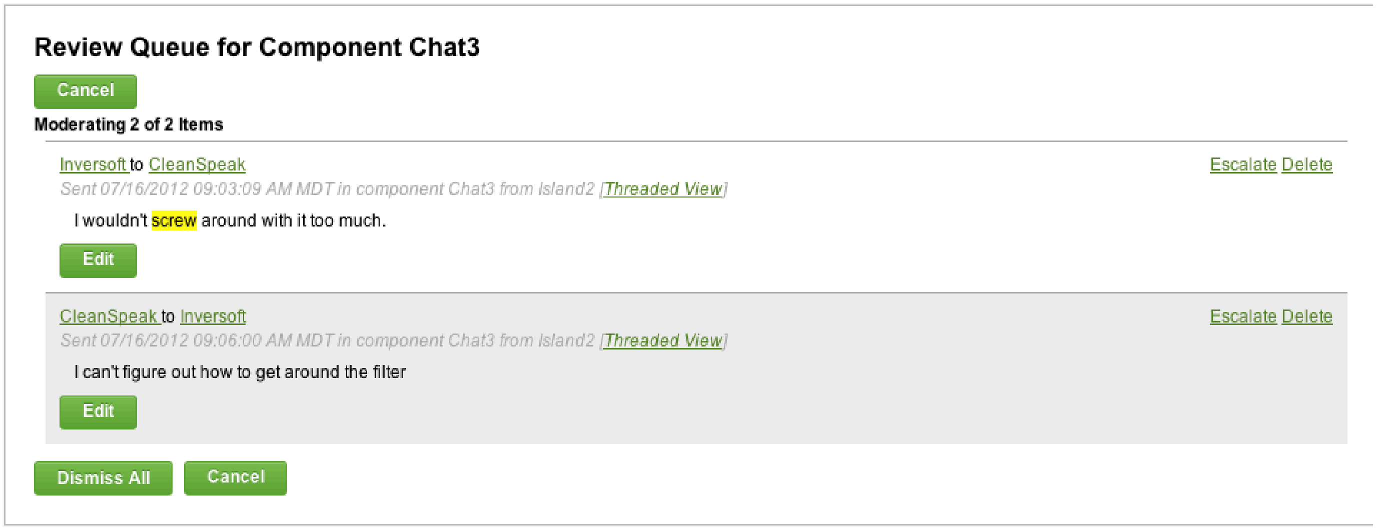
Next Step: Queues

Queues

Queues are used to prioritize content that moderators should review. See the Moderation Concepts page with a description of how to align the Pending Alert, and Pending Approval queues with your business objectives. Also review the Setup & Configuration section to learn how to configure each component for each type of queue.

- Content Display - Alerts
- Content Display - Approvals
- Content Display Order
- Display Name
- Match Highlights

Content Display - Alerts


Review Queue for Component Chat3

Moderating 2 of 2 Items

Inversoft to CleanSpeak
Sent 07/16/2012 09:03:09 AM MDT in component Chat3 from Island2 [Threaded View]

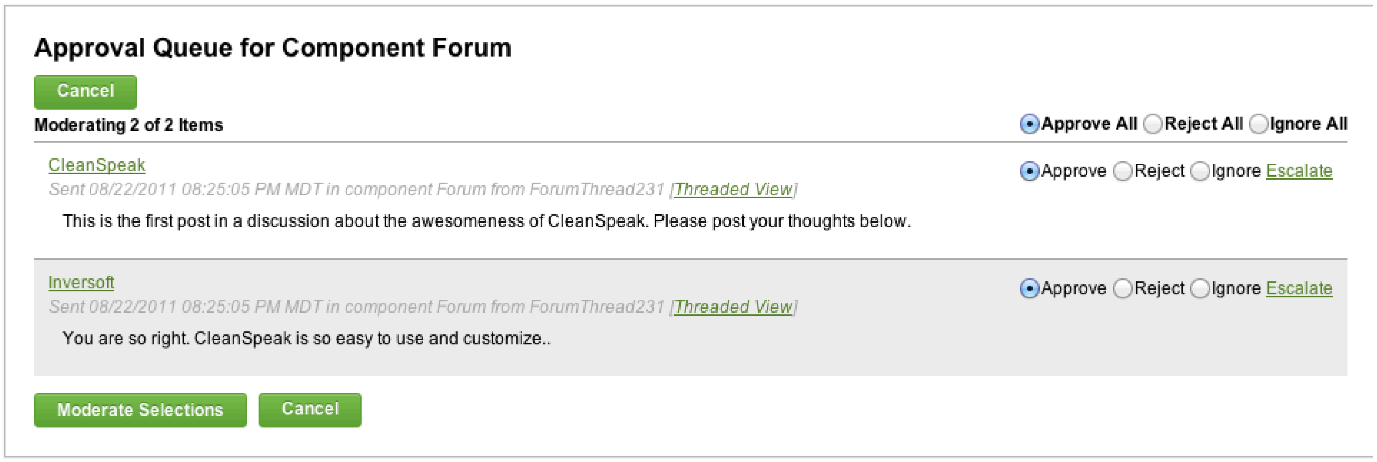
I wouldn't screw around with it too much.

CleanSpeak to Inversoft
Sent 07/16/2012 09:06:00 AM MDT in component Chat3 from Island2 [Threaded View]

I can't figure out how to get around the filter

- **Sender**: CleanSpeak is the user that generated the content. This can be the senderUID or an optional Display Name as described below.
- **Receiver**: Inversoft is the user that received the content. When a receiver is present, this indicates a 1-to-1 direct message.
- **Time**: The time and date the content was generated.
- **Component**: The name of the component where the content is being stored.
- **Location**: Optional value used to view conversations in context. See the Threaded View page for details.
- **Content**: The content that was produced from the Sender. Any part of the content that contains a filter match will be highlighted. See Mat ch Highlights below.
- **Edit**: This button is visible when Content Edit is integrated. See the Edit & Delete page for details.
- **Escalate**: Allows moderators to escalate content to a manger. See the Escalations page for details.
- **Delete**: This link is visible when Content Delete is integrated. See the Edit & Delete page for details.
- **Dismiss All**: Dismiss all content from the queue indicating that the content has been reviewed. The content will be searchable but no longer in a queue.

Content Display - Approvals


Approval Queue for Component Forum

Cancel

Moderating 2 of 2 Items

(● Approve All) Reject All Ignore All

CleanSpeak

Sent 08/22/2011 08:25:05 PM MDT in component Forum from ForumThread231 [Threaded View]

This is the first post in a discussion about the awesomeness of CleanSpeak. Please post your thoughts below.

(● Approve) Reject Ignore Escalate

Inversoft

Sent 08/22/2011 08:25:05 PM MDT in component Forum from ForumThread231 [Threaded View]

You are so right. CleanSpeak is so easy to use and customize..

(● Approve) Reject Ignore Escalate

Moderate Selections Cancel

- **Sender**: CleanSpeak is the user that generated the content. This can be the senderUID or an optional Display Name as described below.
- **Time**: The time and date the content was generated.
- **Component**: The name of the component where the content is being stored.
- **Location**: Optional value used to view conversations in context. See the Threaded View page for details.
- **Content**: The content that was produced from the Sender. Any part of the content that contains a filter match will be highlighted. See Mat ch Highlights below.
- **Moderation Action**:
  - Approve: Approve this content.
  - Reject: Reject this content.
  - Ignore: Ignore the content. Doing so will place the content back into the queue.
  - ***All***: Click the Approve All, Reject All, or Ignore All buttons on the top right of the content display to set all content items to the respective moderation action. This ability may be activated or deactivated for individual moderator accounts from the moderator account page.
- **Escalate**: Allows moderators to escalate content to a manger. See the Escalations page for details.
- **Moderate Selections**: Sends a message to the Notification Server with the moderation action for each piece of content (unless the action is Ignore).

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.

Display Name

Each user in your application has an ID and may also have one or many display names such as character names, forum display names, etc. You may display the ID of the user that usually looks like a long character and/or number sequence such as *user_230263*. Alternatively, you can show a display name. If you would like to see the display name instead of the user ID, inform your development team to assign the additional display name attribute as described in the Moderation WebServices section of the docs.

Match Highlights

When content in a queue contains a filter match, the match will be highlighted. Mousing over the highlighted text will display details of the match.

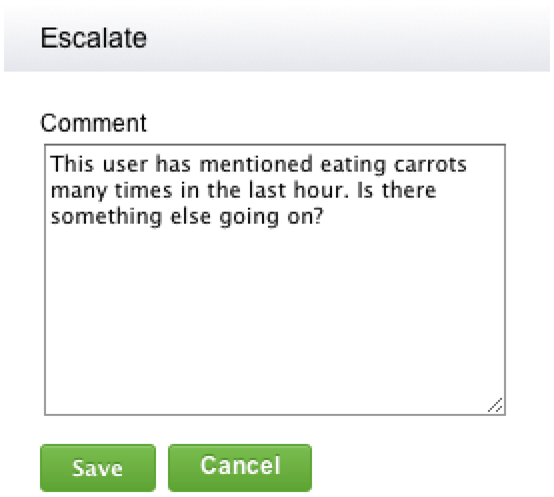
Next Step: Escalations

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. In a case where the moderator would like to review content that has been escalated to a manager, the moderator can search for the content via *Moderation -> Content Search*.

Once content is escalated, a moderation manager can access it by clicking on the Escalations link from the dashboard.



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

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.

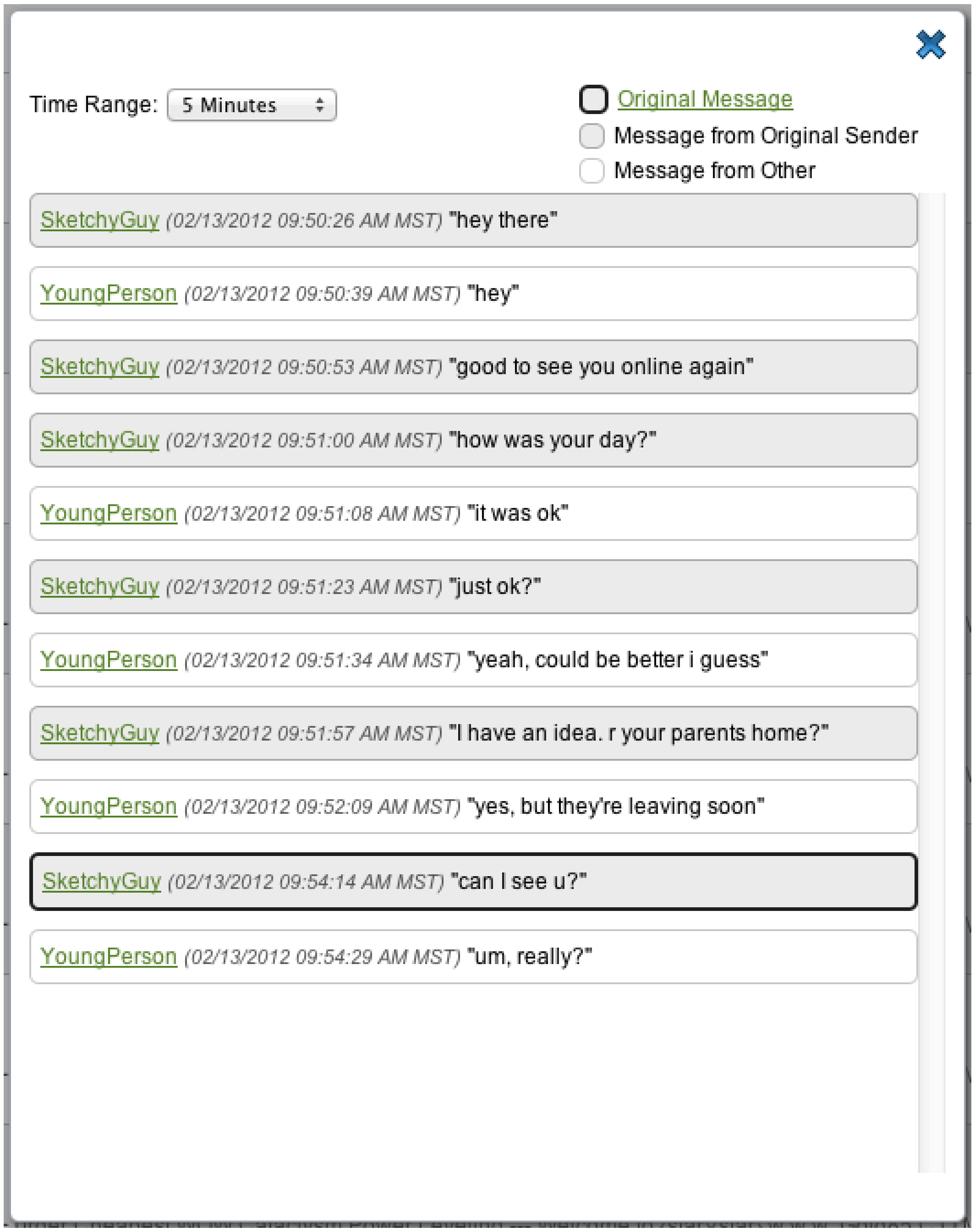
Clicking on a user will open a new tab or window depending on your browser settings so that the user details and history can be reviewed without leaving your queue.

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: 5 Minutes

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

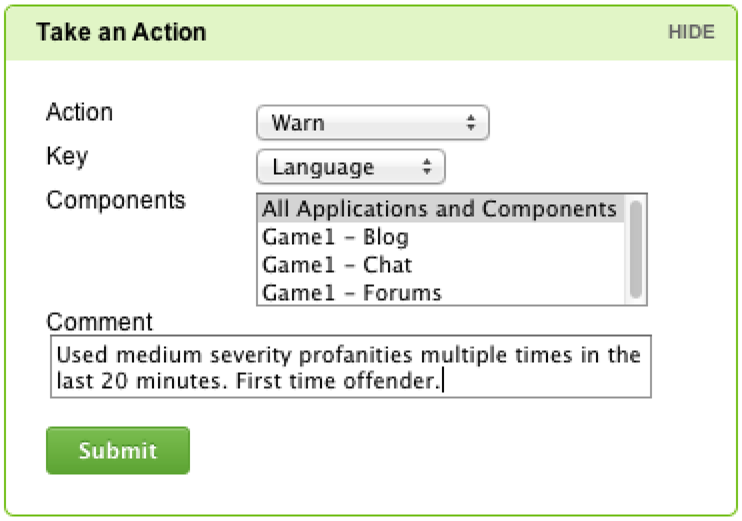
User details and history are available by clicking a user from a content display or result of a User Search. Profile information includes all user data that is sent to CleanSpeak. From the user detail screen, moderators can add comments about the user, review the user's latest content, review all content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

Action: Warn
Key: Language
Components: All Applications and Components, Game1 - Blog, Game1 - Chat, Game1 - Forums
Comment: Used medium severity profanities multiple times in the last 20 minutes. First time offender.

Next Step: Content & User Search

Content & User Search

Comprehensive content and user search forms are available via *Moderation -> Content Search* and *Moderation -> User Search*.

- Content Search
- User Search

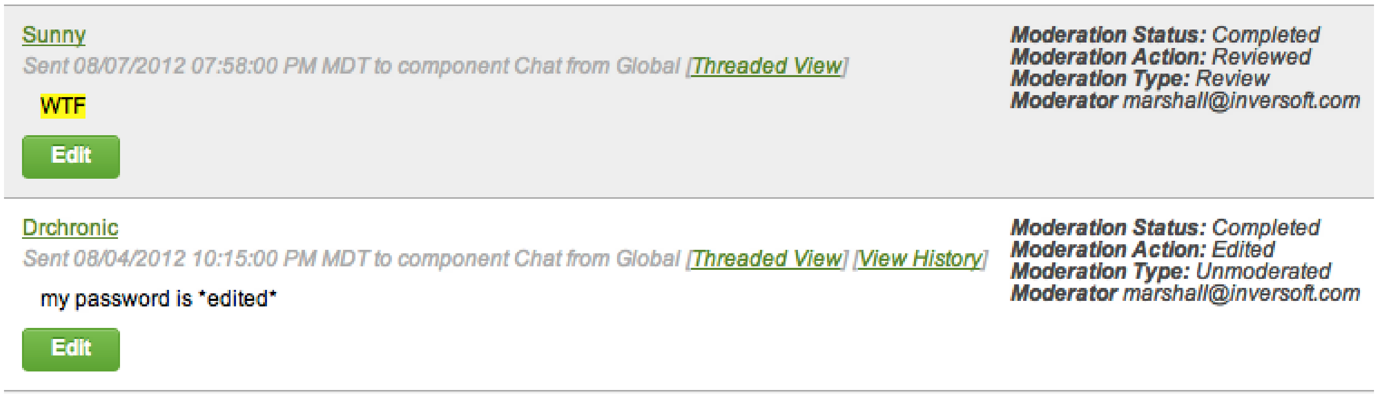
Content Search

Moderation -> Content Search

At least one field must be supplied to narrow down the search results in the form on the left.

Results will be displayed by the time/day the content was submitted by default. Unchecking the Order by Date? check box will display the results that have the higher level severity of filter matches first.

Each result will contain many details including the moderation status, action, type, and the moderator that performed the last action. If the content has prior versions (has been edited), a View History link will be visible. Clicking will pop up a window displaying all versions of the content item.


Sunny
Sent 08/07/2012 07:58:00 PM MDT to component Chat from Global [Threaded View]
WTF

Moderation Status: Completed
Moderation Action: Reviewed
Moderation Type: Review
Moderator marshall@inversoft.com

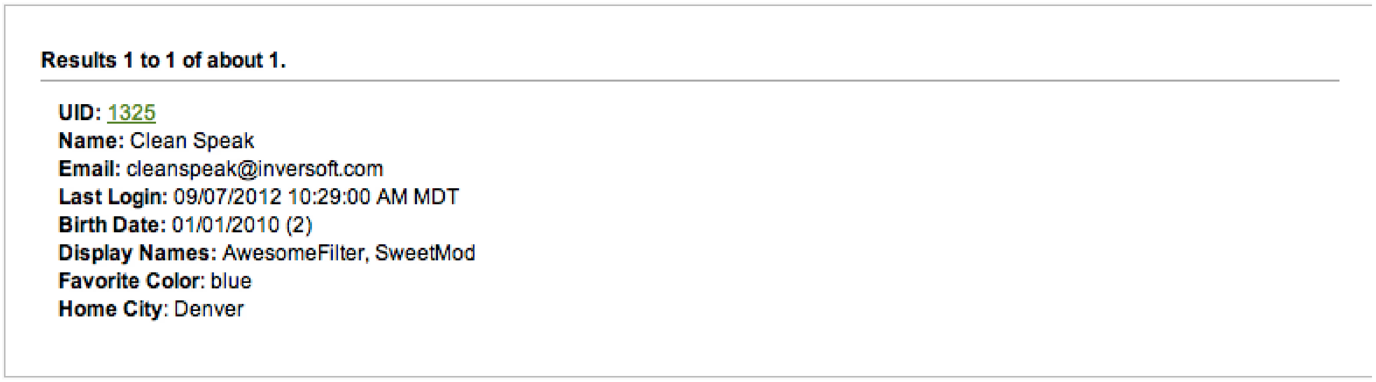
Drchronic
Sent 08/04/2012 10:15:00 PM MDT to component Chat from Global [Threaded View] [View History]
my password is *edited*

Moderation Status: Completed
Moderation Action: Edited
Moderation Type: Unmoderated
Moderator marshall@inversoft.com

User Search

Moderation -> User Search

When the content user service has been integrated, this search form provides the ability to search for users by id, name, email, display name, last login, and/or age. Each result will display a link to the user's profile page.


Results 1 to 1 of about 1.

UID: 1325
Name: Clean Speak
Email: cleanspeak@inversoft.com
Last Login: 09/07/2012 10:29:00 AM MDT
Birth Date: 01/01/2010 (2)
Display Names: AwesomeFilter, SweetMod
Favorite Color: blue
Home City: Denver

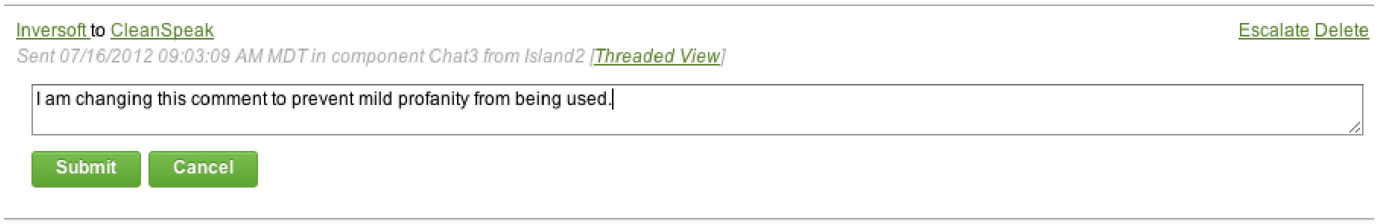
Next Step: Edit & Delete

Edit & Delete

Content sent to CleanSpeak such as forum posts may be edited or deleted within moderation queues or from the content search results page. Each Component must be individually configured to take advantage of the edit/delete feature. Also, additional parameters may need to be sent to CleanSpeak in the code integration as described in the technical documentation: Content Storage & Moderation.

When content is available to be edited, there will be an Edit button below the content. Content may be edited multiple times. When editing is completed and the Submit button is clicked, the content will remain visible in the queue until all content items are dismissed.

For deletable content, there will be a *Delete* link next to the *Escalate* link. When the *Delete* link is clicked, a comment box will be displayed to provide an optional comment of why the content was deleted. Once deleted, the content still exists in your CleanSpeak database even though it is not viewable to the public within your application. You can search for deleted content from the content search page.


Inversoft to CleanSpeak

Sent 07/16/2012 09:03:09 AM MDT in component Chat3 from Island2 [Threaded View]

I am changing this comment to prevent mild profanity from being used.

Next Step: Reporting

Reporting

The CleanSpeak reporting feature is available to clients that have the moderation features licensed and is available from the Reporting link in the main navigation bar.

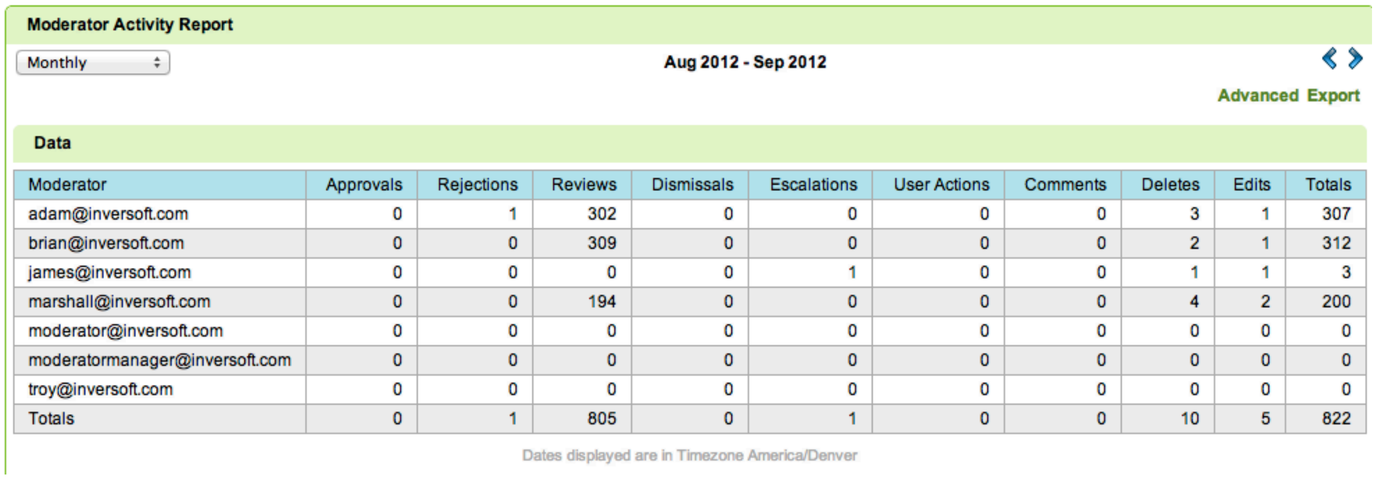
There are two reporting sections: The first provides an overview of moderator activity and the second provides a snapshot of top content producers and top offenders. From either report, you can search by date/time range and export the results to a .csv file. Click on the links below for more detail.

- Moderation Reports
- Top Reports

Moderation Reports

The moderator activity report provides an overview of actions moderators have taken individually and as a whole within a specified time frame. Moderation actions include content dismissed from alert queues, content approved/rejected from approval queues, and more (see the screen shot below).

- Adjusting the Date/Time
- Export
- Charts



Moderator Activity Report

Monthly

Aug 2012 - Sep 2012

Advanced Export

Data

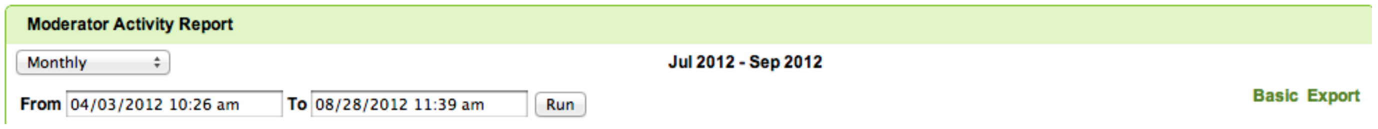
Moderator	Approvals	Rejections	Reviews	Dismissals	Escalations	User Actions	Comments	Deletes	Edits	Totals
adam@inversoft.com	0	1	302	0	0	0	0	3	1	307
brian@inversoft.com	0	0	309	0	0	0	0	2	1	312
james@inversoft.com	0	0	0	0	1	0	0	1	1	3
marshall@inversoft.com	0	0	194	0	0	0	0	4	2	200
moderator@inversoft.com	0	0	0	0	0	0	0	0	0	0
moderatormanager@inversoft.com	0	0	0	0	0	0	0	0	0	0
troy@inversoft.com	0	0	0	0	0	0	0	0	0	0
Totals	0	1	805	0	1	0	0	10	5	822

Dates displayed are in Timezone America/Denver

Adjusting the Date/Time

The select box at the top left of the screen can be used for quick reports segmented quarter hourly, hourly, daily, monthly, or yearly. The arrows to the top right provide the ability to scroll through pages of reports from the previous selection.

Specific time/date ranges may be selected by clicking the Advanced link (see below). The arrows to scroll through pages will be hidden. Clicking into the date range box will display a calendar with hour/minute settings to specify the day/time range of the report.



Moderator Activity Report

Monthly

Jul 2012 - Sep 2012

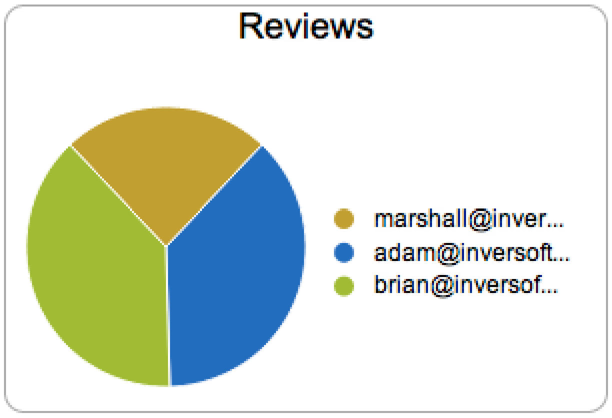
From 04/03/2012 10:26 am To 08/28/2012 11:39 am

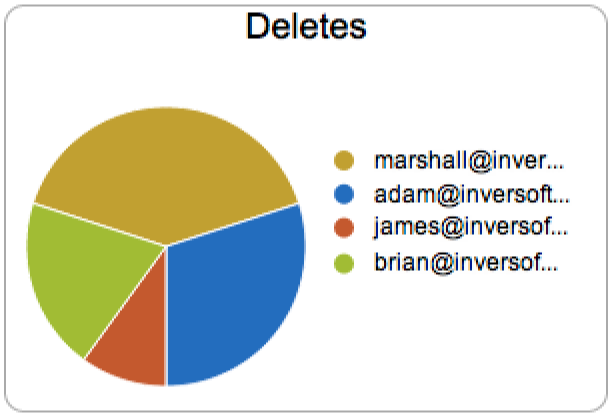
Export

Results displayed by any given time range may be exported to a .csv file by clicking Export at the top right. This will download a file into your browser's downloads folder. It is recommended to rename this file with the time range of the export once the export has completed.

Charts

Pie graphs will be shown for each moderation action when applicable. The graphs are segmented by individual moderators that performed the action at least once by the amount of times the moderator performed that action in relation to the total amount of times the action was performed by all moderators.





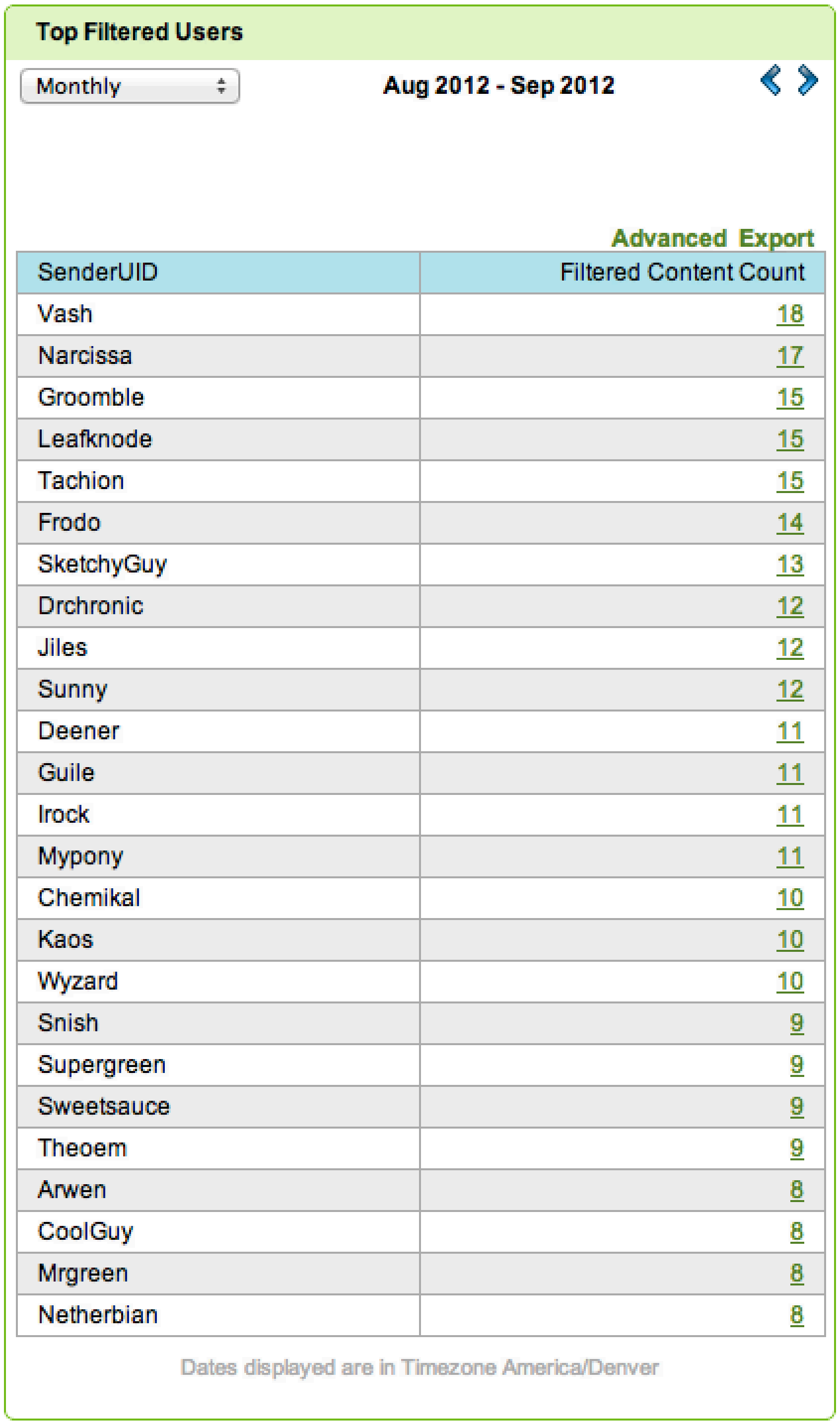
Next Step: Top Reports

Top Reports

Top Reports provides a snapshot of top content users and top offenders (users that submitted content that contained a blacklist filter match).

See the Moderator Reports section for a description of selecting the time frame and exporting the reports.

Each number displayed under Filtered Content Count (or Content Count for the Top Producing Users table) is a link to the Content Search page. Clicking the number will pre populate the Content Search Form with the date range and sender UID who generated the content.



SenderUID	Filtered Content Count
Vash	18
Narcissa	17
Groomble	15
Leafknode	15
Tachion	15
Frodo	14
SketchyGuy	13
Drchronic	12
Jiles	12
Sunny	12
Deener	11
Guile	11
Irock	11
Mypony	11
Chemikal	10
Kaos	10
Wyzard	10
Snish	9
Supergreen	9
Sweetsauce	9
Theoem	9
Arwen	8
CoolGuy	8
Mrgreen	8
Netherbian	8

Dates displayed are in Timezone America/Denver

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 “analleptic” (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.