



## **Cisco SocialMiner Developer Guide Release 10.5(1)**

**First Published:** June 04, 2014

### **Americas Headquarters**

Cisco Systems, Inc.  
170 West Tasman Drive  
San Jose, CA 95134-1706  
USA  
<http://www.cisco.com>  
Tel: 408 526-4000  
800 553-NETS (6387)  
Fax: 408 527-0883

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: <http://www.cisco.com/go/trademarks>. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)

© 2014 Cisco Systems, Inc. All rights reserved.



## CONTENTS

---

### Preface

### Overview xi

What's New in Release 10.5(1) xi

---

### CHAPTER 1

### API Conventions 1

ID Variables 2

changeStamp 3

Passwords 4

HTTP Responses 4

API Authentication 4

Field Constraints and Limitations 5

---

### CHAPTER 2

### Authentication 7

Authentication API Commands 7

GET 7

PUT 8

Authentication API Parameters 8

Enable SSL for Active Directory Authentication 9

Enable SSL for Active Directory on the SocialMiner Server 10

---

### CHAPTER 3

### Bayesian Filter Training 11

Bayesian Filter Training API Commands 11

PUT (Train) 11

DELETE 12

---

### CHAPTER 4

### Callback 13

Callback API Commands 13

POST 14

|                               |    |
|-------------------------------|----|
| GET (Create Callback Request) | 16 |
| GET                           | 17 |
| DELETE                        | 20 |
| Callback API Parameters       | 21 |

---

## CHAPTER 5

### Campaign 23

|                         |    |
|-------------------------|----|
| Campaign API Commands   | 23 |
| POST                    | 23 |
| PUT                     | 24 |
| DELETE                  | 25 |
| GET                     | 25 |
| GET (List)              | 26 |
| GET (Suggested Tags)    | 28 |
| Campaign API Parameters | 28 |

---

## CHAPTER 6

### Campaign Results 31

|                                      |    |
|--------------------------------------|----|
| Campaign Results API Commands        | 31 |
| GET                                  | 31 |
| Campaign Results URL Parameters      | 33 |
| Campaign Results Response Parameters | 34 |

---

## CHAPTER 7

### Campaign Results Count 37

|                                     |    |
|-------------------------------------|----|
| Campaign Results Count API Commands | 37 |
| GET                                 | 37 |

---

## CHAPTER 8

### Chat Feed 39

|  |    |
|--|----|
| Create a Chat Feed                             | 39 |
| POST   | 40 |
| Chat Request Form Submission                   | 40 |
| GET Method                                     | 41 |
| GET (Create Chat Request)                      | 41 |
| Chat Feed API Parameters (Create Chat Request) | 42 |
| GET (Events)                                   | 42 |
| GET (Events) Parameters                        | 43 |
| GET (Chat Transcript)                          | 44 |

[PUT \(Update\)](#) 46  
[PUT \(Leave Chat\)](#) 47  
[DELETE](#) 47  
[Chat Session Timeout](#) 48

---

**CHAPTER 9****[Configuration for Multichannel Routing](#) 49**

[Configuration for Multichannel Routing API Commands](#) 49  
[GET](#) 49  
[PUT](#) 50

---

**CHAPTER 10****[Conversation \(Twitter\)](#) 51**

[Conversation API Commands](#) 51  
[GET \(List\)](#) 51

---

**CHAPTER 11****[Email](#) 55**

[Email API Commands](#) 55  
[GET](#) 55  
[PUT](#) 56  
[Email API Parameters](#) 56

---

**CHAPTER 12****[Facebook Reply](#) 59**

[Facebook Reply API Commands](#) 59  
[GET](#) 59  
[POST \(Comment\)](#) 60  
[POST \(Like\)](#) 61  
[GET \(Like\)](#) 62  
[GET \(User\)](#) 62

---

**CHAPTER 13****[Feed](#) 65**

[Feed API Commands](#) 65  
[POST](#) 65  
[DELETE](#) 66  
[GET \(List\)](#) 66  
[GET](#) 69  
[PUT](#) 70

|   |    |
|---|----|
| Feed API Parameters                         | 71 |
| Authorize Against Twitter Feeds             | 75 |
| Twitter Account Authorization Communication | 75 |
| Twitter authorization APIs                  | 76 |
| oauthGetStatus                              | 76 |
| oauthCancel                                 | 77 |
| oauthCallback                               | 77 |
| Authorize Facebook Accounts                 | 78 |
| Facebook Account Authorization              | 79 |
| Facebook authorization APIs                 | 79 |
| FacebookOAuthGetStatus                      | 79 |
| FacebookOAuthCancel                         | 80 |
| FacebookCallback                            | 80 |

---

## CHAPTER 14

|   |           |
|---|-----------|
| <b>Filter</b>                                 | <b>83</b> |
| Filter API Commands                           | 83        |
| POST  | 83        |
| DELETE  | 84        |
| GET (List)                                    | 84        |
| GET   | 85        |
| GET (Script Content)                          | 86        |
| PUT   | 86        |
| Filter API Parameters                         | 87        |
| About Script Filters                          | 88        |
| Script Binding                                | 89        |
| Develop and Test Script Filters               | 90        |
| Script Filter Security                        | 92        |
| Sample Script Filters                         | 93        |
| Script Filter for Social Contact Modification | 94        |
| Script Filter for Klout Score Generation      | 94        |

---

## CHAPTER 15

|                             |           |
|-----------------------------|-----------|
| <b>Filter Results</b>       | <b>97</b> |
| Filter Results API Commands | 97        |
| GET                         | 97        |

---

**CHAPTER 16****Notification Rule 99**

Notification API Commands 100

POST 100

PUT 101

DELETE 102

GET (List) 102

GET 103

Notification API Parameters 104

Notification Keywords in Email and IM 106

HTTP Notifications 107

---

**CHAPTER 17****Predefined Response 109**

Create Predefined Response 109

Get Predefined Response 110

List Predefined Response 111

List Predefined Response (By Group) 111

Update Predefined Response 112

Delete Predefined Response 112

---

**CHAPTER 18****Proxy 115**

Proxy API Commands 115

GET 115

PUT 116

Proxy API Parameters 116

---

**CHAPTER 19****Public URL Prefix for Chat Invitation 119**

GET 119

PUT 119

---

**CHAPTER 20****Purge 121**

Purge API Commands 121

GET (List) 121

PUT 122

Purge API Parameters 122

---

**CHAPTER 21****Push Feed 125**

Push Feed API Commands 125

POST 125

GET 125

---

**CHAPTER 22****Reply Template 127**

Reply Template API Commands 127

POST 127

DELETE 128

GET (List) 128

GET 129

PUT 129

Reply Template API Parameters 130

---

**CHAPTER 23****Reporting Server 131**

Reporting Server API Commands 131

GET 131

Reporting Server API Parameters 132

---

**CHAPTER 24****Reporting User 133**

Reporting User API Commands 133

POST 133

DELETE 134

GET (List) 134

GET 134

PUT 135

Reporting User API Parameters 135

---

**CHAPTER 25****Serviceability 137**

Serviceability API Commands 137

GET 137

GET (List) 138

Get (Performance Information) 141

Serviceability API Parameters 143



Feed statusDescription Values 153

---

## CHAPTER 26

### Social Contact 155

Social Contact API Commands 155

POST 155

GET 156

PUT (update) 158

GET (Chat Transcript) 159

GET (Search) 160

Social Contact API Parameters 164

---

## CHAPTER 27

### Tag 169

Tag API Command 169

GET (List) 169

---

## CHAPTER 28

### Twitter Reply 171

Twitter Follow API Flow 172

Twitter Reply API Commands 172

GET 173

GET (User) 174

GET (Friendships/Exists) 175

POST (Create Status - Tweet) 175

POST (Create Direct Message) 176

POST (Create Retweet) 177

POST (Create Follow) 178

POST (Create Unfollow) 178

---

## CHAPTER 29

### URL Shortener 181

URL Shortener API Commands 181

POST 181

GET 182

GET (List) 183

PUT (update) 184

DELETE 184

---

**CHAPTER 30****XMPP 187**[XMPP API Commands 187](#)[GET 187](#)[PUT 188](#)[XMPP API Parameters 188](#)

---

**APPENDIX A****Reporting Database Connection 191**[Reporting Database SQL Connection 191](#)[Reporting Database Schema 192](#)

---

**APPENDIX B****SocialMiner Server Configuration 197**[Security Configuration Options 197](#)[Port-Forward Firewall Configuration 197](#)[Reverse Proxy 197](#)

---

**APPENDIX C****XMPP BOSH Eventing 199**[Publish and Subscribe 199](#)[Nodes 200](#)[Events 200](#)

---

**APPENDIX D****Custom reply Templates 203**[Javascript Concepts 203](#)



## Overview

---

This document introduces Application Programming Interface (API) use and conventions for SocialMiner and provides details about each API. It also describes other areas of concern to developers such as XMPP BOSH Eventing, the reporting database, and security configuration options.

- [What's New in Release 10.5\(1\)](#), page xi

## What's New in Release 10.5(1)

The following sections describe changes introduced with this release of SocialMiner.

### Chat Transcript Download

SocialMiner Release 10.5(1) provides a chat transcript download option. Two new REST APIs have been implemented that can be used to download the chat transcript while the chat session is in progress. The transcripts can be downloaded in XML or PDF format. For more information, see [GET \(Chat Transcript\)](#), on page 44.

### Search for Contact by Contact ID

SocialMiner Release 10.5(1) introduces the ability to search for a social contact using the contact ID.

The Social Contact [GET \(Chat Transcript\)](#) API now returns an <id> field in the response payload. This field contains the ID of the chat contact to which the transcript belongs. The field name for the contact ID is de.id. The value for de.id is upper case and case-sensitive.

You can use the [GET \(Search\)](#) API to search for the contact with the de.id.

### Default to SSL for External Server Configurations

SocialMiner now defaults to SSL for external server configurations. For the Email API, the default value for the sslEnabled parameter is now "true". For HTTP notifications, the default value of the sslVerifyCertificates parameter is now "true".

These changes apply only to new installations of SocialMiner or settings that were not previously configured prior to upgrade. If you upgrade from a previous release, server configuration settings are preserved. Settings for existing notifications are also preserved.

**Serviceability API Additions**

The [Serviceability API Commands](#) parameter serviceStates now includes the state of the ActiveMQ JMS server.

**Predefined Responses for Multi-session Chat**

SocialMiner Release 10.5(1) provides Multi-session Chat for Unified Contact Center Express (Unified CCX). A new REST API has been provided to define and manage predefined responses that Unified CCX Multi-session Chat agents can use to reply to customers. For more information, see [Predefined Response](#), on page 109.



## CHAPTER

# 1

## API Conventions

SocialMiner uses [REST](#)-based API functions accessed over http. Five API functions are supported; each is mapped to an http operation or command. Not all functions are used for all components.

The URL format is:

```
http://<ServerIP>:<Port>/ccp-webapp/ccp/<Component>
```

where <ServerIP> is the IP address or hostname of the SocialMiner server and the default port is 8080.

The functions are:

- **create (http POST)**—creates an object in the database and returns a response that contains the URL reference to the newly created object. This example response shows the URL reference returned for a newly-created feed:

*http/1.1 201 Created Location: http://192.168.0.1/ccp-webapp/ccp/feed/100162.*

The id for the feed is *100162*.

This URL reference can be used to retrieve the object with an http GET.



---

**Note** In some APIs (for example, the callback API), you can also create objects with an http GET (create). Although GET (create) does not take a payload, the API developer must supply the required parameters in the URL.

---

- **delete (http DELETE)**—deletes an object.
- **get (http GET)**—returns data for an object. For objects for which there are multiple records, GET takes an identifier variable <id> of some kind. For different APIs, the <id> can appear as a <publicid>, an <objectid>, or another form of id variable.
- **update (http PUT)**—modifies an object. For some objects, PUT must include a [changeStamp](#), but all other parameters are optional. Some parameters cannot be modified with a PUT as the change would impact system integrity. For example, you cannot change Feed Type or Filter Type. A PUT with a modification to these read-only parameters generates an error.
- **list (http GET)**—for objects for which there can be multiple records, returns a list. For different components, GET (list) takes different optional URL parameters that modify the content of the returned list. The optional parameters are defined in the sections for each component.

The POST and PUT operations take a payload for which the input format is XML. GET and DELETE calls do not take a payload.

The content type for all POST and PUT operations is application/xml.

Other than http headers, all output is provided as XML.

XML is case sensitive, therefore all xml element names are case sensitive. For example, <Name> and <name> are two different XML elements.

Boolean values (true and false) are not case sensitive.

If a payload contains duplicate fields, only the first one is interpreted by the server.

- [ID Variables, page 2](#)
- [changeStamp, page 3](#)
- [Passwords, page 4](#)
- [HTTP Responses, page 4](#)
- [API Authentication, page 4](#)
- [Field Constraints and Limitations, page 5](#)

## ID Variables

Different forms of identifier variables are used with the different API functions and components. This section provides a summary of some of the key <id> variables.

### objectId

An *objectId* is the generic term for any identifier generated when using http POST to create objects. However, not all identifiers use this generic form. Campaigns use a *publicId* that has special characteristics described below.

Throughout this document, you will see references to different forms of objectIds (such as callbackFeedId or contactId) that are used to distinguish the different objects being addressed. Unless noted below, consider all <id>s found in this document as objectIds.

The DELETE, GET, and PUT operations are performed using the relevant id in the REST URL. For example:

- Use this URL to view results for a specified filter:  
*http://<ServerIP>:<Port>/ccp-webapp/ccp/filter/<id>/results.*
- Use this URL to delete a feed: *http://<ServerIP>:<Port>/ccp-webapp/ccp/feed/<id>.*
- Use this URL to retrieve data for a single callback contact:  
*https://<ServerIP>:<Port>/ccp/callback/contact/<id>.*

Use the list (GET) function to determine object identifiers. In this example, the id for the Facebook test feed is 111852.

```
<Feeds>
<Feed>
  <authToken>*****</authToken>
  <changeStamp>4</changeStamp>
  <name>Facebook Test</name>
```

```

    <pollingInterval>300</pollingInterval>
    <refURL>http://[ServerIP]:[Port]/ccp-webapp/ccp/feed/111852</refURL>
    <status>0</status>
    <tags/>
    <type>4</type>
    <url>http://www.facebook.com/TestSocialMiner</url>
  </Feed>
</Feeds>

```

### publicId

Using http POST to create a campaign object generates a *publicId* based on the campaign name.



#### Note

The *publicId* is not editable after it is created and does not change if you change the name of the campaign.

You can generate your own *publicId* rather than have SocialMiner generate one based on the campaign name. The *publicId* must conform to [RFC 3986](#) for URL syntax. Spaces, slashes, and backslashes are not allowed in the *publicId*, and it cannot be blank. When SocialMiner creates a *publicId* from the provided campaign name element, the string is formatted according to RFC 3986: spaces are replaced with underscores and slashes or backslashes are replaced with hyphens.

If the encoded name results in a collision with another object of the same type, integers starting at 1 are appended to the encoded name until a non-colliding ID is found. If the user changes the name of the campaign later, the *publicId* will not change.

DELETE, GET, and PUT operations for a campaign are performed using the *<publicId>* in the REST URL. For example:

- Use this URL to get the count of results for a specified campaign:  
*http://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/<publicId>/count).*
- Use this URL to update the configuration of a given campaign:  
*http://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/<publicId>.*

Use the list (GET) function with *summary=false* to see the *publicId* for a campaign.

### progressId

Using http GET for Facebook or Twitter generates a *progressId* for the object. The *progressId* is used to retrieve the status of a Facebook or Twitter API call.

## changeStamp

A *changeStamp* is a required parameter for the body of a PUT (update) operation for these objects:

- feed
- campaign
- filter
- reply template
- notification
- social contact

- predefined response

A changeStamp is returned as part of a read or creation operation. The changeStamp must be included in any modify or delete request (a method known as optimistic locking) to prevent clients from unintentionally overwriting each others data. If you do not provide a changeStamp or the changeStamp is out of date, the update fails.

If the update is successful, the database increments the changeStamp by 1.

## Passwords

For security, the APIs do not return passwords in cleartext. Password elements are masked (\*\*\*\*\*).

## HTTP Responses

All errors are returned as [http 1.1 status codes](#). The common codes used by the APIs are:

- **200 OK:** Success
- **201 Created:** The requested item was created.
- **202 Accepted:** The request was accepted. Generally, a URL is provided to obtain additional details, for example, for polling the OAuth status.
- **400 Bad Request:** The request is invalid. Information returned in the ApiErrors message (the example below) shows more details.
- **401 Unauthorized:** The authentication credentials were not supplied or were incorrect.
- **404 Not Found:** The URI requested does not exist on the server.
- **405 Method Not Allowed:** The method specified in the request line is not allowed for the resource identified by the Request-URI.
- **501 Not Implemented:** The server does not have the functionality to fulfill the request identified by the Request-URI.
- **503 Service Unavailable Error:** The requested operation is unavailable at this time. This error is returned by a chat request when the chat limit has been reached.

Field-specific errors and database errors are provided in an XML error message with the format:

```
<ApiErrors>
  <ApiError>
    <ErrorType>Type of Error</ErrorType>
    <ErrorData>Field Error Occurred</ErrorData>
    <ErrorMessage>A Description of the Error</ErrorMessage>
  </ApiError>
</ApiErrors>
```

## API Authentication

SocialMiner APIs that require authentication are grouped under ccp-webapp. The SocialMiner public APIs that do not require authentication are grouped under ccp.



The username and password credentials that were configured for the administrator during installation are used for http basic authentication for those APIs that require it.

When you submit an API call through a web browser, for example

`http://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/`, the browser prompts for the username and password.

When accessing the API through an application such as cURL or POSTER, you must pass the username and password with the request, as in this example:

```
curl -I -X GET
http://username:password@<ServerIP>:<Port>/ccp-webapp/ccp/campaign/
```

Failing to provide a username and password or providing incorrect credentials returns **401 Unauthorized**.

If you forget the administrator credentials, refer to the [Cisco Systems Command Line Interface \(CLI\) document](#) for commands you can run to reset them.

## Field Constraints and Limitations

All user-visible configuration objects (such as feeds, campaigns, and filters) have name and description fields. These fields share common constraints on size and the number of characters allowed to ensure a consistent user experience. For common fields, the characteristics are:

| Field       | Min length | Max length | Allowed characters   |
|-------------|------------|------------|--|
| Name        | 1          | 85         | All UTF-8 characters except non-printing ASCII (0-31 and 127). |
| Description | 0          | 85         | All UTF-8 characters except non-printing ASCII (0-31 and 127). |

Symbols and special characters are allowed in these fields, but they must be handled carefully (and escaped as required).

See [Provisioning](#) in the *SocialMiner User Guide* for information on limitations.





## Authentication

The authentication API allows you to configure a connection to a Microsoft Active Directory (AD) server. You can specify that all users who exist in AD have access to SocialMiner, or you can specify a single group of AD users.

This API is represented on the SocialMiner user interface by the System Administration panel on the Administration tab.



### Note

Only the administrator created during install can use this API.

- [Authentication API Commands](#), page 7
- [Enable SSL for Active Directory Authentication](#), page 9

## Authentication API Commands

This section describes the supported commands for the authentication API and the parameters for those commands.

### Related Topics

- [GET](#), on page 7
- [PUT](#), on page 8
- [Authentication API Parameters](#), on page 8

## GET

Retrieves the authentication information from SocialMiner.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/authentication/ |
| <b>HTTP method:</b> | GET   |

|                              |  |
|------------------------------|--|
| <b>Example XML response:</b> | <pre> &lt;Authentication&gt;   &lt;enabled&gt;true&lt;/enabled&gt;   &lt;managerDistinguishedName&gt;     CN=administrator,CN=users,DC=cceu-doc-ad,     DC=cisco,DC=com   &lt;/managerDistinguishedName&gt;   &lt;managerPassword&gt;*****&lt;/managerPassword&gt;   &lt;primaryHost&gt;10.xx.yyy.zzz&lt;/primaryHost&gt;   &lt;primaryPort&gt;3268&lt;/primaryPort&gt;   &lt;primaryUseSSL&gt;false&lt;/primaryUseSSL&gt;   &lt;refURL&gt;     http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/     authentication   &lt;/refURL&gt;   &lt;roleName&gt;&lt;/roleName&gt; &lt;/Authentication&gt; </pre> |
| <b>Parameters:</b>           | See <a href="#">Authentication API Parameters</a> , on page 8.   |

## PUT

Updates the authentication information from SocialMiner.

|                             |   |
|-----------------------------|---|
| <b>URL:</b>                 | http://<ServerIP>:<Port>/ccp-webapp/ccp/authentication/   |
| <b>HTTP method:</b>         | PUT   |
| <b>Example XML payload:</b> | <pre> &lt;Authentication&gt;   &lt;enabled&gt;true&lt;/enabled&gt;   &lt;primaryHost&gt;ad.server&lt;/primaryHost&gt;   &lt;primaryPort&gt;3268&lt;/primaryPort&gt;   &lt;primaryUseSSL&gt;false&lt;/primaryUseSSL&gt;   &lt;managerDistinguishedName&gt;     cn=admin,ou=users,dc=ad,dc=server   &lt;/managerDistinguishedName&gt;   &lt;managerPassword&gt;password&lt;/managerPassword&gt;   &lt;roleName&gt;CCP_Users&lt;/roleName&gt; &lt;/Authentication&gt; </pre> |
| <b>Parameters:</b>          | See <a href="#">Authentication API Parameters</a> , on page 8.  |

## Authentication API Parameters

Parameters are optional unless otherwise noted.

| Parameter                       | Description   | Notes   |
|---------------------------------|---|---|
| <b>enabled</b>                  | True/False. Indicates if the authentication settings are used when trying to authenticate a user.                                       | If this is disabled, only the application administrator will have access to the system. |
| <b>managerDistinguishedName</b> | The distinguished name of a user that has manager access to the AD server. For example CN=Administrator, CN=users, DC=MYSERVER, DC=COM. | Required if the enabled parameter is true.  |

| Parameter              | Description   | Notes   |
|------------------------|---|---|
| <b>managerPassword</b> | The password of the user specified in the managerDistinguishedName field. | Required if enabled parameter is true.  |
| <b>primaryPort</b>     | The host port.  | Required if enabled parameter is true.  |
| <b>primaryHost</b>     | The host address of the AD server.  | Required if enabled parameter is true.  |
| <b>primaryUseSSL</b>   | Indicates if a secure connection should be established.                   | This requires that a domain certificate be uploaded to the server and that the primaryPort allows secure connections.<br><br><b>If set to true, then you must follow the instructions in <a href="#">Enable SSL for Active Directory Authentication</a>, on page 9.</b> |
| <b>roleName</b>        | The name of an AD role or group.  | All users in this AD role or group are provided access to SocialMiner. Users in AD who are not members of this role or group are not provided access to SocialMiner. Blank or * indicates that all users in AD are allowed to use the application.                      |

## Enable SSL for Active Directory Authentication

You can enable secure authentication (SSL) against a Microsoft Active Directory server by exchanging the SocialMiner certificate with the AD server.

On the Active Directory Server:

### Procedure

---

- Step 1** Verify that the Active Directory has the Certificate Services service installed.
  - Step 2** Select **All Programs > Administrative Tools > Certification Authority**.
  - Step 3** Expand the domain node and select **Issued Certificates**.
  - Step 4** Double click the certificate to open it.
  - Step 5** Open the Details tab and click **Copy to file**.
  - Step 6** An Export wizard appears. In the wizard, select DER encoded binary.
  - Step 7** Use the wizard to select a location to save the file.
  - Step 8** Click **Finish**.
- 

## Enable SSL for Active Directory on the SocialMiner Server

On the SocialMiner server:

### Procedure

---

- Step 1** Open the *Cisco Unified Operating System Administration* by entering the URL `http://<servername>/cmplatform` or by using the Platform Administration link in the System Administration gadget.
  - Step 2** Select **Security > Certificate Management**.
  - Step 3** Click **Upload Certificate**.
  - Step 4** For the Certificate Name, select **tomcat-trust**.
  - Step 5** In the Upload File field, locate the file to upload by clicking **Browse...** and select the certificate file you saved from the Active Directory server.
  - Step 6** Click **Upload File**.
  - Step 7** Restart the Cisco Tomcat service by using the CLI to run the command `utils service restart Cisco Tomcat`.
-



## Bayesian Filter Training

The Bayesian filter training API allows you to define whether or not social contacts containing specific types of content should be included or excluded from campaigns to which the filter is applied.

- [Bayesian Filter Training API Commands, page 11](#)

## Bayesian Filter Training API Commands

This section describes the supported API commands for the Bayesian filter training API and the parameters for those commands.

### Related Topics

[PUT \(Train\), on page 11](#)

[DELETE, on page 12](#)

## PUT (Train)

Adds the content of a document (text in the REST call) or social contact to the specified filter (objectId) and indicates whether to include or exclude this type of content when the filter is run.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/filter/<objectId>/train   |
| <b>HTTP method:</b> | PUT   |
| <b>Parameters:</b>  | <ul style="list-style-type: none"><li>• <b>document:</b> Required if socialcontact is not specified. String. The text on which to train the filter.</li><li>• <b>socialContact:</b> Required if document is not specified. String. The URL of the social contact containing the content on which to train the filter.</li><li>• <b>match:</b> Required. Boolean.<ul style="list-style-type: none"><li>◦ Set to “True” to include social contacts with similar content in campaigns where the filter is applied.</li></ul></li></ul> |

|   |  |
|---|--|
|   | <ul style="list-style-type: none"> <li>◦ Set to “False” to exclude social contacts with similar content in campaigns where the filter is applied</li> </ul>  |
| <b>Example XML request payload using socialContact:</b> | <pre>&lt;TrainingRequest&gt;   &lt;socialContact&gt;     http://[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/     B83B18F4100001292B3D088D0A568DDE   &lt;/socialContact&gt;   &lt;match&gt;True&lt;/match&gt; &lt;/TrainingRequest&gt;</pre>   |
| <b>Example XML request payload using document:</b>      | <pre>&lt;TrainingRequest&gt;   &lt;document&gt;This is very positive. I really like it. Performance   was excellent.   Great product.&lt;/document&gt;   &lt;match&gt;True&lt;/match&gt; &lt;/TrainingRequest&gt;</pre>  |
| <b>HTTP response headers:</b>                           | <pre>http/1.1 200 OK Pragma: No-cache Cache-Control: no-cache Expires: Wed, 31 Dec 1969 19:00:00 EST Set-Cookie: JSESSIONIDSSO=58AEE69D45227D9FE1704D18F9C72913; Path=/ Set-Cookie: JSESSIONID=98504C52667551FFF276F885628BC3B9; Path=/ccp-webapp Content-Type: text/plain Content-Length: 0 Date: Mon, 14 Jun 2010 14:13:09 GMT Server:</pre> |

**Related Topics**

[ID Variables, on page 2](#)

## DELETE

Deletes all training data for a filter.

|                               |   |
|-------------------------------|---|
| <b>URL:</b>                   | <pre>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/filter/ &lt;objectId&gt;/trainingdata</pre>      |
| <b>HTTP method:</b>           | DELETE  |
| <b>HTTP response headers:</b> | <pre>http/1.1 200 OK Content-Type: text/plain Content-Length: 0 Date: Mon, 14 Jun 2010 14:22:30 GMT</pre> |

**Related Topics**

[ID Variables, on page 2](#)





## Callback

The Callback API allows callback applications to send a notification to CCE for an agent to make a voice call to a customer, at the customer's request. The API works in conjunction with a callback feed, campaigns, and a Connection to CCE Notification.

You can use the API to poll the status of the request (including estimated wait time) and to cancel previous callback requests (see DELETE below).



### Note

Only the callback API can be used to **submit or cancel** callback requests. It cannot be done with the Social contact API.

Before creating a callback request, you must have a Callback feed (type 10) assigned to a campaign (see [Feed, on page 65](#)) and you must have a Connection to CCE notification set up. The notification is triggered by a specific tag that is automatically created in the callback social contact.

The response of the create request contains a URL in the location field that can be used by applications to retrieve the status of the callback request, including the estimated wait time.

The URL is available until the request has been cancelled or until there has been no polling detected for at least five minutes.



### Note

The estimated wait time is calculated only once CCE, so it is not necessary to update that value on each poll.

- [Callback API Commands, page 13](#)
- [Callback API Parameters, page 21](#)

## Callback API Commands

This section describes the supported commands for the Callback API and the parameters for those commands.

# POST

Sends the XML body of a contact to SocialMiner to make a callback request.

Alternatively, GET can be used to create a callback request. GET uses UTF-8 encoded URL parameters to provide the parameters required for the contact. See [GET \(Create Callback Request\)](#), on page 16.



## Note

The variables cv\_1 to cv\_10 are included as call variables when SocialMiner initiates a callback request with CCE. All variables starting with "user\_" are included as expanded call context (ECC) variables when SocialMiner initiates a callback request with CCE.

|  |   |
|--|---|
| <b>URL:</b>  | https://<ServerIP>:<Port>/ccp/callback/feed/<callbackFeedId><br>The <callbackFeedId> specifies the callback feed to target for the callback request.  |
| <b>HTTP method:</b>  | POST  |
| <b>Example XML request payload:</b>  | <pre>&lt;Contact&gt;   &lt;name&gt;name&lt;/name&gt;   &lt;title&gt;title&lt;/title&gt;   &lt;description&gt;description&lt;/description&gt;   &lt;mediaAddress&gt;phoneNumber&lt;/mediaAddress&gt;   &lt;tags&gt;     &lt;tag&gt;tag1&lt;/tag&gt;     &lt;tag&gt;tag2&lt;/tag&gt;   &lt;/tags&gt;   &lt;variables&gt;     &lt;variable&gt;       &lt;name&gt;cv_[1-10]&lt;/name&gt;       &lt;value&gt;callVariableValue&lt;/value&gt;     &lt;/variable&gt;     &lt;variable&gt;       &lt;name&gt;user_(eccVariableName)&lt;/name&gt;       &lt;value&gt;eccVariableValue&lt;/value&gt;     &lt;/variable&gt;     &lt;variable&gt;       &lt;name&gt;anythingElseExtensionFieldName&lt;/name&gt;       &lt;value&gt;anythingElseExtensionFieldValue&lt;/value&gt;     &lt;/variable&gt;   &lt;/variables&gt; &lt;/Contact&gt;</pre> <p><b>Note</b> The contact name, title, and mediaAddress (the phone number to be called) are required to create the request.</p> |
| <b>A reference URL to the contact is returned in the location field in the header:</b> | https://<ServerIP>:<Port>/ccp/callback/contact/6EEF968810000132000015F60A568DFB   |
| <b>Response codes:</b>   | 201 Created<br>400 Bad Request<br>See <a href="#">HTTP Responses</a> for more information about the response codes.   |

**Extension fields, call variables, and ECC variables :**

There is an associated callback contact in SocialMiner for every contact that is processed by the callback API. All of the variables associated with the contact in the create request are included as extension fields in the SocialMiner callback contact. For example, a contact with the following XML:

```
<Contact>
  <name>Customer</name>
  <title>Help</title>
  <mediaAddress>5551212</mediaAddress>
  <variables>
    <variable>
      <name>cv_7</name>
      <value>test7</value>
    </variable>
    <variable>
      <name>user_user.callback.test</name>
      <value>ct7</value>
    </variable>
    <variable>
      <name>location</name>
      <value>Boston, MA</value>
    </variable>
  </variables>
</Contact>
```

results in the following SocialMiner callback contact being created:

```
<SocialContact>
  <author>Customer</author>
  <title>Help</title>
  <description/>
  <extensionFields>
    <extensionField>
      <name>mediaAddress</name>
      <value>5551212</value>
    </extensionField>
    <extensionField>
      <name>location</name>
      <value>Boston, MA</value>
    </extensionField>
    <extensionField>
      <name>cv_7</name>
      <value>test7</value>
    </extensionField>
    <extensionField>
      <name>user_user.callback.test</name>
      <value>ct7</value>
    </extensionField>
    <extensionField>
      <name>ewt</name>
      <value>8</value>
    </extensionField>
  </extensionFields>
  <status>handled</status>
  <statusReason>externally_handled</statusReason>
</SocialContact>
```



#### Note

Not all SocialMiner callback contact attributes have been included in this example.

All variables pulled from contact XML are stored as extension fields in the SocialMiner callback contact. The variables cv\_1 to cv\_10 are treated as call variables. All variables starting with "user\_" are treated as ECC variables. The "cv\_" and "user\_" prefixes are not case sensitive.

This example shows how to set call variable 7 to "test7", ECC variable user\_user.callback.test to "ct7", and the extension field location to Boston.

```
<Contact>
```

```

<name>Customer</name>
<title>Help</title>
<mediaAddress>5551212</mediaAddress>
<variables>
  <variable>
    <name>cv_7</name>
    <value>test7</value>
  </variable>
  <variable>
    <name>user_user.callback.test</name>
    <value>ct7</value>
  </variable>
  <variable>
    <name>location</name>
    <value>Boston</value>
  </variable>
</variables>
</Contact>

```

## GET (Create Callback Request)

As an alternative to POST, GET (create callback request) uses UTF-8 encoded URL parameters to provide the parameters required for the callback contact.



### Note

The variables cv\_1 to cv\_10 are included as call variables when SocialMiner initiates a callback request with CCE. All variables starting with "user\_" are included as ECC variables when SocialMiner initiates a callback request with CCE.

|   |  |
|---|--|
| <b>URL:</b>   | https://<ServerIP>:<Port>/ccp/callback/feed/<callbackFeedId><br>The <callbackFeedId> specifies the callback feed to target for the callback request. |
| <b>HTTP method:</b>   | GET  |
| <b>Parameters:</b>  | See <a href="#">Callback API Parameters</a> .  |
| <b>The contact is returned in the location field in the header:</b> | https://<ServerIP>:<Port>/ccp/callback/contact/6EEF968810000132000015F60A568DFB  |
| <b>Response codes:</b>  | 201 Created<br>400 Bad Request<br>See <a href="#">HTTP Responses</a> for more information about the response codes.                                  |

When using GET to create a callback, variables must be passed as query parameters. Variables are denoted by the "variable\_" prefix. For example, use the following GET command to create a callback contact with

- call variable 7 set to "test7"
- ECC variable user\_user.callback.test set to "ct7"

```

http://sample_server/ccp/callback/feed/12345?
name=Customer
&title=Help

```

```
&mediaAddress=5551212
&variable_cv_7=test7
&variable_user_user.callback.test=ct7
```

**Note**

Sending an ECC variable that is not configured in CCE when creating a callback request will not result in the request failing. (CCE will ignore the variable when processing the request.)

Use the 'tags' parameter in the query string to include tags when creating a callback request using GET.

```
http://sample_host/ccp/callback/feed/12345
?name=Customer
&title=Help
&mediaAddress=5551212
&tags=tag1,tag2,tag3
```

## GET

Returns a reference URL for a single callback contact.

|  |  |
|--|--|
| <b>URL:</b>  | https://<ServerIP>:<Port>/ccp/callback/contact/<ContactID>   |
| <b>HTTP method:</b>  | GET  |
| <b>Parameters:</b>   | See <a href="#">Callback API Parameters</a> .  |
| <b>A reference URL to the contact is returned in the location field in the header:</b> | <p>https://&lt;ServerIP&gt;:&lt;Port&gt;/ccp/callback/contact/6EEF968810000132000015F60A568DFB</p> <p><b>Note</b> The URL returned in the GET call is available until the request is cancelled or until there has been no polling for at least five minutes.</p> |
| <b>Response codes:</b>   | <p>200 OK</p> <p>400 Bad Request</p> <p>For more information about the response codes, see <a href="#">HTTP Responses, on page 4</a>.</p>  |

**Note**

The customer callback application developer is responsible for the messages that their customer callback interface provides to the customer. The <status> and <statusReason> fields can be used to understand the state of the contact.

Contact transition states during callback flow are listed in the table below.

| Contact state | State reason code  | Notes  |
|---------------|--------------------|--|
| Queued        | EXTERNALLY_HANDLED | The callback request was successfully submitted to the contact center for routing. |

| Contact state | State reason code  | Notes   |
|---------------|--------------------|---|
| Handled       | EXTERNALLY_HANDLED | The callback request was successfully routed to a contact center agent. |

| Contact state | State reason code                           | Notes   |
|---------------|---|---|
| Discarded     | <b>StatusReason for failure</b>             |   |
|               | NOTIFICATION_INVALID_NEW_TASK_MESSAGE       | CCE routing error: Invalid message in route request.  |
|               | NOTIFICATION_MEDIA_ROUTING_DISABLED         | CCE routing error: Routing is disabled.   |
|               | NOTIFICATION_NO_SCRIPT                      | CCE routing error: No available script to run.<br>Mis-configuration of CCE. No script is available to run.  |
|               | NOTIFICATION_INVALID_MRD_ID                 | CCE routing error: Invalid media routing domain.<br>Mis-configuration of CCE. Invalid CCE Media Routing Domain.<br>Ensure that the Media Routing Domain configured in the SocialMiner Notification exists in the CCE configuration.                               |
|               | NOTIFICATION_INVALID_SCRIPT_SELECTOR        | CCE routing error: Invalid dialed number or script selector.<br>Mis-configuration of CCE. Invalid CCE Dialed Number/Script Selector.<br>Ensure that the Dialed Number/Script Selector configured in the SocialMiner Notification exists in the CCE configuration. |
|               | NOTIFICATION_ROUTER_RELEASED_TASK           | CCE routing error: CCE released the call or task.<br>The CCE script indicated that the callback request should be dropped. Possible script mis-configuration.   |
|               | NOTIFICATION_UNKNOWN_ROUTING_PROBLEM        | CCE routing error: Unknown problem.   |
|               | NOTIFICATION_CCE_CONNECTION_LOST            | The connection to CCE was lost or was not established.  |
|               | NOTIFICATION_CCE_SOCIALMINER_SYSTEM_FAILURE | SocialMiner failed (or restarted) while the callback was queued.  |
|               | NOTIFICATION_INVALID_VARIABLE               |   |

| Contact state | State reason code         | Notes   |
|---------------|---------------------------|---|
|               |                           | <p>SocialMiner could not submit the task to CCE because it contained invalid media address, call variable, or ECC variable values. The maximum field lengths for these fields are</p> <ul style="list-style-type: none"> <li>• media address: 39 bytes</li> <li>• call variable: 40 bytes</li> <li>• ECC variable name: 32 bytes</li> <li>• ECC variable value: 210 bytes.</li> </ul> |
|               | NOTIFICATION_RATE_LIMITED | The incoming rate of callback contacts has exceeded 40 contacts per minute.   |

## DELETE

Cancels a callback request. Once the DELETE request is successfully processed (response code 200 is received by the application), the application cannot use GET to poll for the status of the contact.



### Note

It is possible for the customer to receive a callback after cancelling the callback request if the cancellation request arrives after an agent was already selected.

|                                     |   |
|-------------------------------------|---|
| <b>URL:</b>                         | https://<ServerIP>:<Port>/ccp/callback/contact/<Id>   |
| <b>HTTP method:</b>                 | DELETE  |
| <b>Example request XML payload:</b> | None  |
| <b>Response codes:</b>              | <p>200 OK</p> <p>400 Bad Request</p> <p>See <a href="#">HTTP Responses</a> for more information about the response codes.</p> |

The following table shows the statusReason codes that can be triggered by the cancellation requests for callback contacts in different states.

| Social contact state | StatusReason for cancel request            | Meaning                                |
|----------------------|--|--|
| QUEUED               | NOTIFICATION_CCE_CALLBACK_CANCEL_REQUESTED | Cancel callback request was initiated. |



| Social contact state | StatusReason for cancel request            | Meaning   |
|----------------------|--|---|
| DISCARDED            | NOTIFICATION_CCE_CALLBACK_CANCEL_SUCCEEDED | CCE cancelled the callback successfully.  |
| HANDLED              | EXTERNALLY_HANDLED                         | The cancellation request failed because the task was routed to an agent before the callback could be cancelled. |

## Callback API Parameters

Parameters are optional unless otherwise noted.

| Parameter                | Description  | Notes  |
|--------------------------|--|--|
| <b>title</b>             | The name of the callback request.  | Required for POST and GET (create).  |
| <b>name</b>              | The person who generated the callback request.   | Required for POST and GET (create).<br>The name is limited to a maximum of 100 characters and a newline (\n) is not allowed in the name. |
| <b>mediaAddress</b>      | The phone number to be called.   | Required for POST and GET (create).  |
| <b>description</b>       | An optional description to accompany the callback request.   |  |
| <b>tags</b>              | One or more tags or keywords associated with the contact.  | To include tags when using GET to create a callback contact, use the parameter 'tags' in the query string.                               |
| <b>variables</b>         | A set of custom name and value pairs.  | When using GET to create a callback, variables must be passed as query parameters and start with 'variable_'.                            |
| <b>status</b>            | The status of a callback request.  | See "Contact State" in <a href="#">GET, on page 17</a> .   |
| <b>estimatedWaitTime</b> | An estimate of the amount of time (in seconds) until an agent's phone will be available to place the call. |  |





## Campaign

The Campaign API allows you to create, update, delete, get, and list campaigns in the system.

Campaigns are collections of feeds (see [Feed](#)) and filters (see [Filter](#)) that generate lists of results matching the criteria defined in the campaign.

This API is represented on the SocialMiner user interface by the Campaigns panel on the Configuration tab.

- [Campaign API Commands](#), page 23

## Campaign API Commands

This section describes the supported commands for the Campaign API and the parameters for those commands.

### Related Topics

- [POST](#)
- [PUT](#), on page 24
- [DELETE](#), on page 25
- [GET](#)
- [GET \(List\)](#)
- [GET \(Suggested Tags\)](#), on page 28
- [Campaign API Parameters](#), on page 28

## POST

Creates a campaign.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | <a href="http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/campaign">http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/campaign</a> |
| <b>HTTP method:</b> | POST  |
| <b>Parameters:</b>  | See <a href="#">Campaign API Parameters</a> , on page 28.   |

|                                     |   |
|-------------------------------------|---|
| <b>Example XML request payload:</b> | <pre> &lt;Campaign&gt;   &lt;name&gt;MyTestCampaign&lt;/name&gt;   &lt;publicId&gt;MyTestCampaign&lt;/publicId&gt;   &lt;description&gt;This is my test campaign&lt;/description&gt;   &lt;includeExpr&gt;Cisco Expert Advisor&lt;/includeExpr&gt;   &lt;excludeExpr&gt;ICM&lt;/excludeExpr&gt;   &lt;chatInvitationFeed&gt;     http://[ServerIP]:[Port]/ccp-webapp/ccp/feed/5000   &lt;/chatInvitationFeed&gt;   &lt;feeds&gt;     &lt;feed&gt;       http://[ServerIP]:[Port]/ccp-webapp/ccp/feed/5000     &lt;/feed&gt;     &lt;feed&gt;       http://[ServerIP]:[Port]/ccp-webapp/ccp/feed/5001     &lt;/feed&gt;   &lt;/feeds&gt;   &lt;filters&gt;     &lt;filter&gt;       http://[ServerIP]:[Port]/ccp-webapp/ccp/filter/6000     &lt;/filter&gt;     &lt;filter&gt;       http://[ServerIP]:[Port]/ccp-webapp/ccp/filter/6001     &lt;/filter&gt;   &lt;/filters&gt; &lt;/Campaign&gt; </pre> |
| <b>HTTP response headers:</b>       | <p>If the campaign is successfully created, the URL of the created resource is returned.</p> <pre> http/1.1 201 Created Location: http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/campaign/MyTestCampaign Content-Type: text/plain Content-Length: 0 Date: Tue, 12 Jan 2010 16:41:14 GMT </pre> <p>See also <a href="#">HTTP Responses</a>.</p>   |

## PUT

Updates an existing campaign.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | <p>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/campaign/&lt;publicId&gt;</p> <p>For more information about &lt;publicId&gt;, see <a href="#">ID Variables</a>, on page 2.</p> |
| <b>HTTP method:</b> | PUT   |
| <b>Parameters:</b>  | See <a href="#">Campaign API Parameters</a> , on page 28.   |

**Example XML request payload:**

```

<Campaign>
  <name>MyTestCampaign</name>
  <publicid>MyTestCampaign</publicid>
  <description>This is my test campaign</description>
  <includeExpr>Cisco Expert Advisor</includeExpr>
  <excludeExpr>ICM</excludeExpr>
  <chatInvitationFeed>
    http://[ServerIP]:[Port]/ccp-webapp/ccp/feed/5000
  </chatInvitationFeed>
  <feeds>
    <feed>
      http://[ServerIP]:[Port]/ccp-webapp/ccp/feed/5000
    </feed>
    <feed>
      http://[ServerIP]:[Port]/ccp-webapp/ccp/feed/5001
    </feed>
  </feeds>
  <filters>
    <filter>
      http://[ServerIP]:[Port]/ccp-webapp/ccp/filter/6000
    </filter>
    <filter>
      http://[ServerIP]:[Port]/ccp-webapp/ccp/filter/6001
    </filter>
  </filters>
  <changeStamp>8</changeStamp>
</Campaign>

```

## DELETE

Deletes an existing campaign.

|                               |   |
|-------------------------------|---|
| <b>URL:</b>                   | http://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/<publicId><br>For more information about <publicId>, see <a href="#">ID Variables</a> , on page 2. |
| <b>HTTP method:</b>           | DELETE  |
| <b>HTTP response headers:</b> | http/1.1 200 OK<br>Content-Type: text/plain<br>Content-Length: 0<br>Date: Tue, 12 Jan 2010 17:03:54 GMT   |

## GET

Returns the data for a single campaign.

|                     |  |
|---------------------|--|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/<publicId>?metrics=<true/false><br>For more information about <publicId>, see <a href="#">ID Variables</a> , on page 2. |
| <b>HTTP method:</b> | GET  |
| <b>Parameters:</b>  | See <a href="#">Campaign API Parameters</a> , on page 28.  |

|                               |   |
|-------------------------------|---|
| <b>Example XML response:</b>  | <p><b>Note</b> The “ refURL ” is a copy of the URL requested.</p> <pre> &lt;Campaign&gt;   &lt;refURL&gt;     http://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/MyTestCampaign   &lt;/refURL&gt;   &lt;resultsURL&gt;     http://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/MyTestCampaign/results   &lt;/resultsURL&gt;   &lt;suggestedTagsURL&gt;     http://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/MyTestCampaign/suggestedtags   &lt;/suggestedTagsURL&gt;   &lt;publicid&gt;MyTestCampaign&lt;/publicid&gt;   &lt;description&gt;This is my test campaign&lt;/description&gt;   &lt;includeExpr&gt;Cisco Expert Advisor&lt;/includeExpr&gt;   &lt;excludeExpr&gt;ICM&lt;/excludeExpr&gt;   &lt;chatInvitationFeed&gt;     http://[ServerIP]:[Port]/ccp-webapp/ccp/feed/5000   &lt;/chatInvitationFeed&gt;   &lt;changeStamp&gt;12345&lt;/changeStamp&gt;   &lt;feeds&gt;     &lt;feed&gt;       http://[ServerIP]:[Port]/ccp-webapp/ccp/feed/5000     &lt;/feed&gt;     &lt;feed&gt;       http://[ServerIP]:[Port]/ccp-webapp/ccp/feed/5001     &lt;/feed&gt;   &lt;/feeds&gt;   &lt;filters&gt;     &lt;filter&gt;       http://[ServerIP]:[Port]/ccp-webapp/ccp/filter/6000     &lt;/filter&gt;     &lt;filter&gt;       http://[ServerIP]:[Port]/ccp-webapp/ccp/filter/6001     &lt;/filter&gt;   &lt;/filters&gt;   &lt;metrics&gt;     &lt;socialContactCount&gt;12&lt;/socialContactCount&gt;   &lt;/metrics&gt; &lt;/Campaign&gt; </pre> <p>The metrics element contains &lt;socialContactCount&gt;, which is the number of social contacts associated with this campaign .</p> |
| <b>HTTP response headers:</b> | <pre> http/1.1 200 OK Content-Type: application/xml Transfer-Encoding: chunked Date: Tue, 12 Jan 2010 16:55:05 GMT </pre>   |

## GET (List)

Lists all configured campaigns.

|             |  |
|-------------|--|
| <b>URL:</b> | <pre> http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/campaign?summary=&lt;true/false&gt; </pre> <p>Where <i>summary</i> is an optional query parameter that is false by default.</p> <p>OR</p> <pre> http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/campaign?metrics=&lt;true/false&gt; </pre> |
|-------------|--|

|                               |  |
|-------------------------------|--|
|                               | Where <i>metrics</i> is an optional query parameter that is false by default   |
| <b>HTTP method:</b>           | GET  |
| <b>Example XML responses:</b> | <p>When summary is true and metrics is false:</p> <pre> &lt;Campaigns&gt;   &lt;Campaign&gt;     &lt;refURL&gt;       http://[ServerIP]:[Port]/ccp-webapp/ccp/       campaign/MyTestCampaign     &lt;/refURL&gt;   &lt;/Campaign&gt;   &lt;Campaign&gt;     &lt;refURL&gt;       http://[ServerIP]:[Port]/ccp-webapp/ccp/       campaign/MyTestCampaign2     &lt;/refURL&gt;   &lt;/Campaign&gt; &lt;/Campaigns&gt; </pre> <p>When summary is false and metrics is false:</p> <pre> &lt;Campaigns&gt;   &lt;Campaign&gt;     &lt;refURL&gt;       http://[ServerIP]:[Port]/ccp-webapp/ccp/       campaign/MyTestCampaign     &lt;/refURL&gt;     &lt;name&gt;MyTestCampaign&lt;/name&gt;     &lt;publicid&gt;MyTestCampaign&lt;/publicid&gt;     &lt;description&gt;This is my test campaign&lt;/description&gt;     ...   &lt;/Campaign&gt;   &lt;Campaign&gt;     &lt;refURL&gt;       http://[ServerIP]:[Port]/ccp-webapp/ccp/       campaign/MyTestCampaign2     &lt;/refURL&gt;     &lt;name&gt;MyTestCampaign2&lt;/name&gt;     &lt;publicid&gt;MyTestCampaign2&lt;/publicid&gt;     &lt;description&gt;This is my test campaign&lt;/description&gt;     ...   &lt;/Campaign&gt; &lt;/Campaigns&gt; </pre> <p>When summary is false and metrics is true:</p> <pre> &lt;Campaigns&gt;   &lt;Campaign&gt;     &lt;refURL&gt;       http://[ServerIP]:[Port]/ccp-webapp/ccp/       campaign/MyTestCampaign     &lt;/refURL&gt;     &lt;name&gt;MyTestCampaign&lt;/name&gt;     &lt;publicid&gt;MyTestCampaign&lt;/publicid&gt;     &lt;description&gt;This is my test campaign&lt;/description&gt;     &lt;metrics&gt;       &lt;socialContactCount&gt;12&lt;/socialContactCount&gt;     &lt;/metrics&gt;     ...   &lt;/Campaign&gt;   &lt;Campaign&gt;     &lt;refURL&gt;       http://[ServerIP]:[Port]/ccp-webapp/ccp/ </pre> |

```

    campaign/MyTestCampaign2
  </refURL>
  <name>MyTestCampaign2</name>
  <publicid>MyTestCampaign2</publicid>
  <description>This is my test campaign</description>
  <metrics>
    <socialContactCount>12</socialContactCount>
  </metrics>
  ...
</Campaign>
</Campaigns>
Summary set to true and metrics set to true is an invalid combination.

```

## GET (Suggested Tags)

Retrieves the suggested tags for social contacts in a specific campaign. Up to ten tags are returned based on how recent and how often a tag has been used in this campaign.

|                              |  |
|------------------------------|--|
| <b>URL:</b>                  | http://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/<publicId>/suggestedtags<br>For more information about <publicId>, see <a href="#">ID Variables</a> , on page 2.  |
| <b>HTTP method:</b>          | GET  |
| <b>Example XML response:</b> | The first 10 suggested tags are returned<br><br><pre> &lt;SuggestedTags&gt;   &lt;tags&gt;     &lt;tag&gt;tag1&lt;/tag&gt;     &lt;tag&gt;tag2&lt;/tag&gt;     &lt;tag&gt;tag3&lt;/tag&gt;   &lt;/tags&gt; &lt;/SuggestedTags&gt; </pre> |

## Campaign API Parameters

Parameters are optional unless otherwise noted.

| Parameter                 | Description   | Notes   |
|---------------------------|---|---|
| <b>changeStamp</b>        | The change stamp of the campaign record.  | Integer. Defaults to 0.<br>Required for PUT.<br>For more information, see <a href="#">changeStamp</a> .               |
| <b>chatInvitationFeed</b> | The chat invitation feed for the campaign (must be a chat feed); must be set to a feed's reference URL. | A chat invitation feed is required to invite Twitter and Facebook users to chat from the SocialMiner reply templates. |
| <b>description</b>        | The description of the campaign.  |   |
| <b>excludeExpr</b>        | The searching expression to exclude.  |   |



| Parameter                                     | Description   | Notes  |
|---|---|--|
| <b>feeds</b>                                  | A list of feeds linked to the campaign.                         |  |
| <b>filters</b>                                | A list of filters linked to the campaign.                       |  |
| <b>includeExpr</b>                            | The searching expression to include.                            |  |
| <b>metrics</b><br><b>(socialContactCount)</b> | URL Parameter for GET and GET list.                             | True/false. Defaults to false.<br>If “true”, a count of the social contacts in each campaign is returned.<br>If “false”, no social contact count is returned.                  |
| <b>name</b>                                   | The name of the campaign.                                       | Required for POST.   |
| <b>publicId</b>                               | URL-encoded version of name. Must be unique within object type. |  |
| <b>refURL</b>                                 | A copy of the URL requested.                                    |  |
| <b>suggestedTags</b>                          | Up to ten suggested tags for the campaign.                      |  |
| <b>summary</b>                                | URL Parameter for List.   | True/false. Defaults to false.<br>If “true”, only the URLs of the objects are returned.<br>If “false”, full object information is returned along with the URLs of the objects. |





## Campaign Results

The Campaign results API allows you to get the results for a campaign.

This API is represented on the SocialMiner user interface in the Home tab.

- [Campaign Results API Commands, page 31](#)

## Campaign Results API Commands

This section describes the supported command (GET) for the campaign results API and the parameters for that command.

### Related Topics

[GET, on page 31](#)

[Campaign Results URL Parameters, on page 33](#)

[Campaign Results Response Parameters, on page 34](#)

## GET

Gets results for the specified campaign based on an optional index.

|                        |  |
|------------------------|--|
| <b>URL:</b>            | <code>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/campaign/&lt;publicId&gt;/results</code><br>For more information about <publicId>, see <a href="#">ID Variables, on page 2</a> . |
| <b>HTTP method:</b>    | GET  |
| <b>URL parameters:</b> | See <a href="#">Campaign Results URL Parameters, on page 33</a> .  |

**Example response:**

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/
Atom" xmlns:ccp="http://www.cisco.com/
ccbu/ccp/xml/socialcontact/1.0/"
xmlns:dc="http://purl.org/dc/elements/1.1/">

  <title>Business News</title>

  <link rel="self" href="http://[ServerIP]:[Port]/
ccp-webapp/ccp/campaign/Business_News/
results?timestamp=1278696169003"/>
  <link rel="countsince" href="http://192.168.0.1/
ccp-webapp/ccp/campaign/Business_News/count?
postId=22E00F5310000129460A1EB40A568DDE" />
  <link rel="next" href="http://192.168.0.1/
ccp-webapp/ccp/campaign/JingCampaign1/
results?timestamp=1276525157775&startIndex=50" />

  <subtitle>This feed has been created by the Cisco
Customer Collaboration Platform</subtitle>

  <id>http://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/
Business_News/results</id>
  <updated>2010-06-10T17:20:46Z</updated>
  <dc:date>2010-06-10T17:20:46Z</dc:date>
  <stentry>
    <title>SEC OKs 'flash crash' fix</title>
    <link rel="alternate" href="http://rss.cnn.com/~r/
rss/money_latest/~3/h0MxRXFew9I/index.htm" />
    <link rel="socialcontact" href="http://192.168.0.1/
ccp-webapp/ccp/socialcontact/
22E00F5310000129460A1EB40A568DDE" />
    <author>
      <name />
    </author>
    <id>http://rss.cnn.com/~r/rss/money_latest/~3/
h0MxRXFew9I/index.htm</id>
    <updated>2010-06-10T17:10:50Z</updated>
    <published>2010-06-10T17:10:50Z</published>

    <summary type="html">The Securities and Exchange
Commission approved new rules Thursday that will halt
trading uniformly across all U.S. markets for
stocks experiencing wild price swings to prevent
a repeat of last month's "flash crash."<br/>

    </summary>
    <dc:creator />
    <dc:date>2010-06-10T17:10:50Z</dc:date>
    <ccp:scstatustimestamp>
      1283819058417
    </ccp:scstatustimestamp>
    <ccp:scstatus>reserved</ccp:scstatus>
    <ccp:scstatususerid>admin</ccp:scstatususerid>
    <ccp:sctags>
      <ccp:sctag>sometag</ccp:sctag>
      <ccp:sctag>anothertag</ccp:sctag>
      <ccp:sctag>yetanothertag</ccp:sctag>
    </ccp:sctags>
  </stentry>
  <stentry> ... </stentry>
</feed>
```

For more information about the response parameters, see [Campaign Results Response Parameters](#), on page 34.

## Campaign Results URL Parameters

Parameters are optional unless otherwise noted.

| Parameter                    | Description   | Notes   |
|------------------------------|---|---|
| <b>filterStatus</b>          | Display contacts whose status matches a status within this field.             | <p>String. Defaults to all if the parameter is not specified.</p> <p>You must specify a value for the parameter if the parameter is included or no contacts are returned. Can be one or more of (case-insensitive):</p> <ul style="list-style-type: none"> <li>• RESERVED</li> <li>• HANDLED</li> <li>• DRAFT</li> <li>• UNREAD</li> <li>• DISCARDED</li> <li>• QUEUED</li> </ul> <p>• Multiple status example:<br/> <a href="http://192.168.0.1/ccp-webapp/ccp/campaign/Business_News/results?filterStatus=RESERVED&amp;filterStatus=HANDLED">http://192.168.0.1/ccp-webapp/ccp/campaign/Business_News/results?filterStatus=RESERVED &amp; filterStatus=HANDLED</a> </p> |
| <b>filterTag</b>             | Display contacts whose filters matches one or more of the tags in this field. | <p>String. Defaults to all tags if not specified. Example:</p> <p><a href="http://192.168.0.1/ccp-webapp/ccp/campaign/Business_News/results?filterTag=tag1&amp;filterTag=tag2">http://192.168.0.1/ccp-webapp/ccp/campaign/Business_News/results?filterTag=tag1 &amp; filterTag=tag2</a></p>   |
| <b>includePostWithPostId</b> | Specifies whether or not to display contacts with ID equal to postId.         | <p>This option is ignored if postId is not specified. If includePostWithPostId is not specified or is specified and is set to false, the contact with ID equal to postId is not included in the campaign results. If includePostWithPostId is specified and is set to true, the contact with ID equal to postId is included in the campaign results.</p>  |

| Parameter             | Description   | Notes   |
|-----------------------|---|---|
| <b>postId</b>         | The identifier of the userid that made the post.      | String. If provided, results are displayed starting after the provided postId. It cannot be used if timestamp or startIndex is specified.   |
| <b>timestamp</b>      | A given time and date.                                | Integer. Displays results older than this timestamp.<br><br>Defaults to the time of request if not provided. If startIndex is not specified, then timestamp assumes startIndex = 0.   |
| <b>resultsPerPage</b> | The maximum number of results to be returned.         | Integer.<br><br>Default is 50 and maximum is 200.   |
| <b>startIndex</b>     | The number of results to skip based on the timestamp. | Integer. Used for pagination.<br><br>Assuming resultsPerPage is set at the default of 50, you could create a "page 2" link by using the timestamp provided in the href of the <code>feed/link rel="self"</code> and a startIndex of 50. Page 3 would use the same timestamp and a startIndex of 100, and so on. |



#### Note

- If timestamp is provided and startIndex is not provided, then the results are displayed up to the "resultsPerPage" with a creation date older than "timestamp", starting at index 0.
- If timestamp is not provided and startIndex is provided, then the results are displayed up to the "resultsPerPage" with a creation date older than "now" starting at startIndex.
- If postId is provided, then the contact identified by the postId is used as the basis for the search. The social contact for the provided postId does not appear in the results.

## Campaign Results Response Parameters

Results are returned as an [ATOM 1.0](#) feed that can contain the following elements:

| Parameter                                 | Description                                 | Notes |
|---|---|-------|
| <b>feed/title</b>                         | The name of the campaign.                   |       |
| <b>feed/link rel = self &amp; feed/id</b> | The URL of the results that were requested. |       |

| Parameter  | Description  | Notes   |
|--|--|---|
| <b>feed/link rel = countsince</b>                | The URL for the API call for the number of new contacts since this result was retrieved.   | See <a href="#">Campaign Results Count</a> , on page 37.  |
| <b>feed/link rel = next</b>                      | The URL to the next 50 results.  | Present only when there are more than 50 results left in the campaign.<br>Represented in the SocialMiner user interface by the More button. |
| <b>entry/title</b>                               | The title of the contact.  |   |
| <b>entry/link rel = alternate &amp; entry/id</b> | The URL to the contact.  |   |
| <b>entry/link rel = socialcontact</b>            | The URL for the API call of this contact.  |   |
| <b>entry/summary</b>                             | The content of the contact.  |   |
| <b>entry/published</b>                           | The date and time that the contact was published.<br><br>If the contact did not contain a published date, this is the date when the contact was read by SocialMiner. | Date and time form is <i>YYYY-MM-DDTHH:MM:SSZ</i> .   |
| <b>entry/ccp:statustimestamp</b>                 | The timestamp of the last change to the status of the contact.   |   |
| <b>entry/ccp:status</b>                          | The current status of the contact.   |   |
| <b>entry/ccp:scstatusreason</b>                  | The reason the contact changed to its current status.  |   |
| <b>entry/ccp:sctags/ccp:sctag</b>                | One or more tags associated with this contact.   |   |
| <b>entry/ccp:scstatususerid</b>                  | The last user to change the status of this contact.<br><br>If blank and the status is unread, then this contact has never had a status change.                       |   |
| <b>entry/ccp:sourcetype</b>                      | The type of feed that generated or fetched the contact.  |   |

| Parameter                | Description  | Notes |
|--------------------------|--|-------|
| entry/ccp:scissoftlocked | Whether or not the contact state can be modified using the SocialMiner user interface. |       |





## Campaign Results Count

The Campaign results count API allows you to get a count of the results for a specified campaign. You can get a count of the results for the entire campaign or the count of results since a given post.

- [Campaign Results Count API Commands](#), page 37

## Campaign Results Count API Commands

This section describes the supported command (GET) for the campaign results API.

### GET

Gets the number of results in a campaign.

|                              |  |
|------------------------------|--|
| <b>URL:</b>                  | <code>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/campaign/<br/>&lt;publicId&gt;/count</code><br>For more information about <publicId>, see <a href="#">ID Variables</a> , on page 2.  |
| <b>HTTP method:</b>          | GET  |
| <b>URL parameter:</b>        | <b>postId:</b> Optional. The postId of the last post seen. This id is the unique id of the campaign result. A link to this API with the appropriate id in the url is included in the results atom feed. The count displays the number of results published after the referenced campaign result. If no postId is provided, the full number of results in the campaign is returned. The postId can be found in the campaign result, for example:<br><br><code>&lt;link rel="countsince" href="http://[ServerIP]:[Port]/<br/>ccp-webapp/ccp/campaign/MyTestCampaign/count?<br/>postId=92517B6610000128295CEBB40A568DDE" /&gt;</code> |
| <b>Example XML response:</b> | <code>&lt;count&gt;44&lt;/count&gt;</code>   |

 GET



## Chat Feed

---

Chat feeds are used by the Chat REST API. The Chat REST API supports operations on the URL `http://<ServerIP>:<Port>/ccp/chat..`

For more information on feeds, see [Feed](#), on page 65.

- [Create a Chat Feed](#), page 39
- [GET \(Events\)](#), page 42
- [GET \(Chat Transcript\)](#), page 44
- [PUT \(Update\)](#), page 46
- [PUT \(Leave Chat\)](#), page 47
- [DELETE](#), page 47
- [Chat Session Timeout](#), page 48

## Create a Chat Feed

Before creating a chat request, you must have a chat feed assigned to a campaign. After a chat feed is set up, three methods are used to create chat requests. Use the following steps to create the feed and assign it to a campaign:

### Procedure

---

- Step 1** Use the [POST](#) to create a type 8 chat feed.
  - Step 2** Confirm that the POST returned a 201 (created) response code, then look in the location field of the http response header for the reference URL (refURL) of the feed just created.
  - Step 3** Add the chat feed to a campaign. You can create a new campaign and then use the PUT API to add the feed to it or to any existing campaign.
-

## POST

After creating the chat feed, POST the body of the contact to the chat proxy (with no re-direct to the customer chat UI).

|   |  |
|---|--|
| <b>URL:</b>   | http://<ServerIP>:<Port>/ccp/chat  |
| <b>HTTP method:</b>   | POST   |
| <b>Example XML request payload:</b>                               | <pre>&lt;SocialContact&gt;   &lt;feedRefURL&gt;http://[ServerIP]:[Port]/     ccp-webapp/ccp/feed/134268&lt;/feedRefURL&gt;   &lt;author&gt;Test_author&lt;/author&gt;   &lt;title&gt;Social contact title&lt;/title&gt; &lt;/SocialContact&gt;</pre>   |
| The contact is returned in the location field in the header:      | http://<ServerIP>:<Port>/ccp-webapp/ccp/socialcontact/6EEF968810000132000015F60A568DFB   |
| The chat room is in the Extension Fields of the content response: | <pre>&lt;SocialContact&gt;   &lt;author&gt;Test_author&lt;/author&gt;   &lt;createdDate&gt;1316121187977&lt;/createdDate&gt;   &lt;description/&gt;   &lt;extensionFields&gt;     &lt;extensionField&gt;       &lt;name&gt;chatRoom&lt;/name&gt;       &lt;value&gt;socialminer_chat.3@conference.127.0.0.1&lt;/value&gt;     &lt;/extensionField&gt;   &lt;/extensionFields&gt;   &lt;id&gt;6EEF968810000132000015F60A568DFB&lt;/id&gt;   &lt;link&gt;     http://[ServerIP]:[Port]/       ccp-webapp/ccp/socialcontact/       6EEF968810000132000015F60A568DFB   &lt;/link&gt;   &lt;publishedDate&gt;1316121187976&lt;/publishedDate&gt;   &lt;refURL&gt;     http://[ServerIP]:[Port]/       ccp-webapp/ccp/socialcontact/       6EEF968810000132000015F60A568DFB   &lt;/refURL&gt;   &lt;replyToId/&gt;   &lt;status&gt;unread&lt;/status&gt;   &lt;statusTimestamp&gt;1316121187976&lt;/statusTimestamp&gt;   &lt;statusUserId/&gt;   &lt;tags/&gt;   &lt;title&gt;Social contact title&lt;/title&gt; &lt;/SocialContact&gt;</pre> |
| <b>Parameters:</b>  | See <a href="#">Chat Feed API Parameters (Create Chat Request)</a> , on page 42.   |
| <b>HTTP response headers:</b>                                     | See <a href="#">HTTP Responses</a> .   |

## Chat Request Form Submission

Chat requests are created when a customer fills in and submits a chat request form. This method creates the contact.

If the contact is successfully created, the feed is redirected to the customer chat UI. The URL to submit the GET is `http://<ServerIP>:<Port>/ccp/chat/<feedid>/redirect`.

The code below is an example of how to submit a chat request through a form post. Users can customize this basic HTML to suit their needs.

```
<style type='text/css'>span { display: inline-block; width: 100px; }</style>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<form action='https://10.86.141.242/ccp/chat/form/100525' method='post'>
  <span>Title:</span><input id='title' type='text' name='title' /><br/>
  <span>Author:</span><input id='author' type='text' name='author' /><br/>
  <span>Description:</span><input id='description' type='text' name='description' /><br/>

  <span>Tags:</span><input id='tags' type='text' name='tags' /><br/>
  <span>Remarks:</span><input id='remarks' type='text' name='extensionField_remarks'
    value='sample value' /><br/>
  <input id='submit' type='submit' value='Submit' />
  <input type="hidden" name="extensionField_chatLogo" value="./img/ciscoLogoColor.png">
  <input type="hidden" name="extensionField_chatWaiting"
    value="Welcome, please wait while we connect you with a customer care representative.">

  <input type="hidden" name="extensionField_chatAgentJoinTimeOut"
    value="All customer care representatives are busy assisting other clients.
    Please continue to wait or try again later.">
  <input type="hidden" name="extensionField_chatError"
    value="Sorry, the chat service is currently not available. Please try again later.">
</form>
```

See [Chat Feed API Parameters \(Create Chat Request\)](#), on page 42.

## GET Method

Alternately, a simple GET can be used to create a chat request where all the required parameters for the social contact and the chat session created are provided as UTF-8 encoded URL parameters.

If the contact is successfully created, the feed is redirected to the customer chat UI. The URL to submit the GET is `http://<ServerIP>:<Port>/ccp/chat/<feedid>/redirect`.

### GET (Create Chat Request)

Creates a social contact with the chat session for the particular chat feed id (<id>) and redirects the feed to the customer chat UI.

|                              |   |
|------------------------------|---|
| <b>URL:</b>                  | <code>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp/chat/&lt;id&gt;/redirect</code>  |
| <b>HTTP method:</b>          | GET   |
| <b>Example http request:</b> | <code>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp/chat/&lt;id&gt;redirect?title=Test_Title&amp;author=Test_Author</code> |
| <b>Parameters:</b>           | See <a href="#">Chat Feed API Parameters (Create Chat Request)</a> , on page 42.                                      |
| <b>Response:</b>             | 302 redirect.   |

Once submitted, the client needs to continue the chat for the current browser session using the GET/PUT and DELETE operations on URL `http://<ServerIP>:<Port>/ccp/chat` (without the feedid).

## Chat Feed API Parameters (Create Chat Request)

Parameters are optional unless otherwise noted.

| Parameter             | Description   | Notes  |
|-----------------------|---|--|
| <b>title</b>          | The name of the contact.                                | Required   |
| <b>author</b>         | The author of the contact.                              | Required   |
| <b>description</b>    | A description of the content or context of the contact. |  |
| <b>tags</b>           | One or more tags that are associated with this contact. |  |
| <b>extensionField</b> | A custom name/value pair.                               | The name is formatted (extensionField_<field name>).<br>All parameters in the URL must be UTF-8 encoded. |

The remaining APIs in this chapter are used after a chat session has been created.

## GET (Events)

This API gets events queued on the chat proxy starting from the specified eventid. GET returns an XML payload wrapped in a <ChatEvents> tag containing 0 or more event types returned in the order they were received. When first called, eventid should be set to 0. Subsequent calls should identify the id of the last processed event.



### Note

Because there is no identifier specifying which chat we are getting events for, API calls only work with a valid session cookie. The session cookie returned from the POST or GET should be provided on subsequent chat proxy API calls.

|                     |   |
|---------------------|---|
| <b>URL example:</b> | http://<ServerIP>:<Port>/ccp/chat?eventid=0&all=false |
| <b>HTTP method:</b> | GET   |
| <b>Parameter:</b>   | See below.  |

|                                  |   |
|----------------------------------|---|
| <b>Example response payload:</b> | <pre> &lt;ChatEvents&gt;   &lt;PresenceEvent&gt;     &lt;id&gt;2&lt;/id&gt;     &lt;from&gt;Steve&lt;/from&gt;     &lt;status&gt;joined&lt;/status&gt;   &lt;/Presence&gt;   &lt;MessageEvent&gt;     &lt;id&gt;3&lt;/id&gt;     &lt;from&gt;Steve&lt;/from&gt;     &lt;body&gt;Hi There&lt;/body&gt;   &lt;/Message&gt;   &lt;MessageEvent&gt;     &lt;id&gt;4 &lt;/id&gt;     &lt;from&gt;Steve&lt;/from&gt;     &lt;body&gt;How can I help you?&lt;/body&gt;   &lt;/Message&gt; &lt;/ChatEvents&gt; </pre> |
| <b>Responses:</b>                | 200 (Succeeded) or 400/500 (Failed).<br>See <a href="#">HTTP Responses</a> .  |

## GET (Events) Parameters

GET takes two parameters:

- **all** (default = false)—if true, all events since eventid are returned. If false, MessageEvents sent from the consumer (by the PUT call) are omitted from the returned events list.

For example: `http://<ServerIP>:<Port>/ccp/chat?eventid=0&all=false`

- **eventid** (default = 0)— When first called, eventid is set to 0. Subsequent calls identify the eventid of the last processed event. The call will return events that occurred since the specified eventid. It is normal for there to be gaps in the eventids that are returned.

| Event         | Description and child events  |
|---------------|---|
| MessageEvent  | A chat message sent from the agent: <ul style="list-style-type: none"> <li>• id: sequential id of the event.</li> <li>• from: sender of the message.</li> <li>• body: URL-encoded body of the message.</li> </ul> |
| PresenceEvent | A user joined or left the session: <ul style="list-style-type: none"> <li>• id: sequential id of the event.</li> <li>• from: user who joined/left the room.</li> <li>• status: joined left.</li> </ul>            |

| Event       | Description and child events   |
|-------------|--|
| StatusEvent | <p>Chat session status:</p> <ul style="list-style-type: none"> <li>• id: sequential id of the event.</li> <li>• status: status information. <ul style="list-style-type: none"> <li>◦ chat_finished: the chat is finished with no problems (not currently used).</li> <li>◦ chat_finished_error: the chat is finished due to an error; this is not recoverable (not currently used).</li> <li>◦ chat_issue: there is an issue with the chat session; this may be recoverable.</li> <li>◦ chat_ok: the chat session is ok again.</li> </ul> </li> <li>• detail: If the XMPP Server fails, a StatusEvent will be sent with status set to <i>chat_issue</i> and detail set to <i>XMPP Server is down</i>.</li> </ul> |

## GET (Chat Transcript)

### XML Transcript Download

This API retrieves the transcript for a chat contact, which can be downloaded in XML format.



#### Note

Because there is no identifier to specify which chat you are getting events for, API calls only work with a valid session cookie. You can obtain the session ID (JSession ID) from the session and provide it in this request.

|                              |  |
|------------------------------|--|
| <b>URL:</b>                  | http://<ServerIP>:<Port>/ccp/chat/transcript.xml   |
| <b>HTTP method:</b>          | GET  |
| <b>Example XML response:</b> | <pre>&lt;ChatTranscript&gt;&lt;startDate&gt;1389540103973&lt;/startDate&gt; &lt;transcript&gt; &lt;chat&gt;&lt;time&gt;1389540103973&lt;/time&gt;&lt;name&gt;Agent&lt;/name&gt;&lt;msg&gt;Hi&lt;/msg&gt;&lt;/chat&gt; &lt;chat&gt;&lt;time&gt;1389540108058&lt;/time&gt;&lt;name&gt;Dave&lt;/name&gt;&lt;msg&gt;Hello&lt;/msg&gt;&lt;/chat&gt; &lt;chat&gt;&lt;time&gt;1389540111001&lt;/time&gt;&lt;name&gt;Agent&lt;/name&gt;&lt;msg&gt;How may I   help you?&lt;/msg&gt;&lt;/chat&gt; &lt;chat&gt;&lt;time&gt;1389540114026&lt;/time&gt;&lt;name&gt;Dave&lt;/name&gt;&lt;msg&gt;I have a   query&lt;/msg&gt;&lt;/chat&gt; &lt;/transcript&gt; &lt;/ChatTranscript&gt;</pre> |
| <b>Elements:</b>             | <p><b>time:</b> timestamp of the chat in GMT.</p> <p><b>name:</b> agent or customer name.</p> <p><b>msg:</b> the chat message.</p>   |



|                               |   |
|-------------------------------|---|
| <b>HTTP response headers:</b> | 400 Bad Request<br>404 Not Found<br>501 Not Implemented<br>See <a href="#">HTTP Responses, on page 4</a> for more information about the response codes. |
|-------------------------------|---|

### PDF Transcript Download

This API retrieves the transcript for a chat contact, which can be downloaded as a PDF file.

|                               |   |
|-------------------------------|---|
| <b>URL:</b>                   | http://<ServerIP>:<Port>/ccp/chat/transcript.pdf?locale=<locale>  |
| <b>HTTP method:</b>           | GET   |
| <b>HTTP response headers:</b> | 200 OK<br>400 Bad Request<br>404 Not Found<br>501 Not Implemented<br>See <a href="#">HTTP Responses, on page 4</a> for more information about the response codes. |

The file name of the downloaded PDF is in the following format:

chatTranscript\_<Customer\_Name\_spaces\_replaced\_by\_underscores>\_DD\_MMM\_YYYY\_HH\_MM\_in\_24\_hr\_format.pdf

For example, chatTranscript\_John\_Doe\_23\_Aug\_2014\_15\_34.pdf.



#### Note

- The time zone for all time values mentioned in the PDF is the time zone of the deployed SocialMiner server. The time zone is also clearly displayed with each time value.
- To use the locale, you must install the language pack in SocialMiner. For more information about installing language packs, see the *Cisco SocialMiner User Guide*, available here: <http://www.cisco.com/c/en/us/support/customer-collaboration/socialminer/products-user-guide-list.html>

If you do not have the language pack installed, the locale defaults to en\_ALL.

The following table lists the strings that you can use for the locale parameter with their associated languages.

| String | Language |
|--------|----------|
| da_DK  | Danish   |
| de_DE  | German   |
| en_ALL | English  |

| String | Language        |
|--------|-----------------|
| es_ES  | Spanish         |
| fi_FI  | Finnish         |
| fr_CA  | French (Canada) |
| fr_FR  | French (France) |
| it_IT  | Italian         |
| nb_NO  | Norwegian       |
| nl_NL  | Dutch           |
| pl_PL  | Polish          |
| pt_BR  | Portuguese      |
| ru_RU  | Russian         |
| sv_SE  | Swedish         |
| tr_TR  | Turkish         |

**Important**

The following locales/languages are not supported for the PDF output:

- Chinese Simplified (zh\_CN)
- Chinese Traditional (zh\_TW)
- Japanese (ja\_JP)
- Korean (ko\_KR)

## PUT (Update)

This API sends a chat event to the chat room. Currently, the only chat event that can be sent from the client is a chat message of the form:

```
<Message>
  <body>body of message </body>
</Message>
```

**Note**

Because there is no identifier to specify which chat we are getting events for, API calls only work with a valid session cookie. The session cookie returned from the POST or GET should be provided on subsequent chat proxy API calls.

|                                  |  |
|----------------------------------|--|
| <b>URL:</b>                      | http://<ServerIP>:<Port>/ccp/chat  |
| <b>HTTP method:</b>              | PUT  |
| <b>Example response payload:</b> | <pre>&lt;ChatEvents&gt;   &lt;PresenceEvent&gt;     &lt;id&gt;1&lt;/id&gt;     &lt;from&gt;Steve&lt;/from&gt;     &lt;status&gt;joined&lt;/status&gt;   &lt;/Presence&gt;   &lt;MessageEvent&gt;     &lt;id&gt;2&lt;/id&gt;     &lt;from&gt;Steve&lt;/from&gt;     &lt;body&gt;Hi There&lt;/body&gt;   &lt;/Message&gt;   &lt;MessageEvent&gt;     &lt;id&gt;3&lt;/id&gt;     &lt;from&gt;Steve&lt;/from&gt;     &lt;body&gt;How can I help you?&lt;/body&gt;   &lt;/Message&gt; &lt;/ChatEvents&gt;</pre> |
| <b>Responses:</b>                | 200 (Succeeded) or 400/500 (Failed).<br>See <a href="#">HTTP Responses</a> .   |

## PUT (Leave Chat)

This API enables the chat customer to end an ongoing chat. Invoking this API removes the customer from the chat room and as a result, the chat agent is left alone in the chat room. However, the client session between the customer's browser and the SocialMiner server continues to be active until the configured inactivity timeout is reached. This enables the customer to download a transcript for the chat using the transcript API before all session times out.

|                               |   |
|-------------------------------|---|
| <b>URL:</b>                   | http://<ServerIP>:<Port>/ccp/chat/leaveChat   |
| <b>HTTP method:</b>           | PUT   |
| <b>HTTP response headers:</b> | 200 OK<br>400 Bad Request<br>For more information about the response codes, see <a href="#">HTTP Responses, on page 4</a> . |

## DELETE

Stops the chat session.

**Note**

Because there is no identifier to specify which chat we are getting events for, API calls only work with a valid session cookie. The session cookie returned from the POST or GET should be provided on subsequent chat proxy API calls.

|                               |                                      |
|-------------------------------|--------------------------------------|
| <b>URL:</b>                   | http://<ServerIP>:<Port>/ccp/chat    |
| <b>HTTP method:</b>           | DELETE                               |
| <b>HTTP response headers:</b> | See <a href="#">HTTP Responses</a> . |

## Chat Session Timeout

A chat session will time out after five minutes of customer inactivity. If there are no UPDATES or GETS for five minutes, the session is terminated. This timeout value is configurable; five minutes is the default value.



## Configuration for Multichannel Routing

This API configures SocialMiner to listen for a connection from the media routing peripheral gateway (MR PG).

SocialMiner will only allow MR PG(s) from the configured list to connect.

- [Configuration for Multichannel Routing API Commands, page 49](#)

## Configuration for Multichannel Routing API Commands

This section describes the supported commands for the configuration for multichannel routing API and the parameters for those commands.

### Related Topics

[GET, on page 49](#)

[PUT, on page 50](#)

## GET

Get the MR PG configuration.

|                     |  |
|---------------------|--|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/contactcenter/mrconfig/default |
| <b>HTTP method:</b> | GET  |

|                              |   |
|------------------------------|---|
| <b>Example XML response:</b> | <pre>&lt;MRconfig&gt;   &lt;enabled&gt;true&lt;/enabled&gt;   &lt;port&gt;38001&lt;/port&gt;   &lt;hostA&gt;myhostname.abc.com&lt;/hostA&gt;   &lt;hostB&gt;ccx.host.xyz.com&lt;/hostB&gt;   &lt;refURL&gt;     http://socialminer.server.ip/ccp-webapp/     ccp/contactcenter_mrconfig/default   &lt;/refURL&gt; &lt;/MRconfig&gt;</pre> <p>See also <a href="#">API Conventions</a>, on page 1.</p> |
|------------------------------|---|

## PUT

Edit the MR PG connection configuration.

|                                     |  |
|-------------------------------------|--|
| <b>URL:</b>                         | http://<ServerIP>:<Port>/ccp-webapp/ccp/contactcenter/mrconfig/default   |
| <b>HTTP method:</b>                 | PUT  |
| <b>Input and output format:</b>     | XML  |
| <b>Example XML request payload:</b> | <pre>&lt;MRconfig&gt;   &lt;enabled&gt;&gt;false&lt;/enabled&gt;   &lt;port&gt;38001&lt;/port&gt;   &lt;hostA&gt;myhostname.abc.com&lt;/hostA&gt;   &lt;hostB&gt;ccx.host.xyz.com&lt;/hostB&gt; &lt;/MRconfig&gt;</pre>  |
| <b>Parameters:</b>                  | <ul style="list-style-type: none"> <li>• <b>enabled:</b> a Boolean flag to enable or disable the MR configuration. <ul style="list-style-type: none"> <li>◦ If false: MR PG configuration is ignored and no connections are accepted.</li> <li>◦ If true and no hostnames are set: any connection is accepted.</li> <li>◦ If true and at least one hostname is specified: only connections matching the specified hosts are allowed.</li> </ul> </li> <li>• <b>hostA, hostB:</b> host identifiers of the MR PG servers. Any identifier can be specified (for example, IPv4 or hostname), as long as it can be resolved to the actual IP address of the server(s). The combined length of the host strings is limited to 254 characters.</li> <li>• <b>port:</b> the port number that the MR PG uses to connect to SocialMiner. Defaults to 38001. The valid range is 10000 - 65535.</li> </ul> |



## Conversation (Twitter)

The Conversation API allows you to view the conversational context of a specific Twitter social contact. A conversation will be displayed starting at the selected tweet. The selected contact is checked for a reply-to ID and, if there is one, the contact corresponding to the reply-to ID is included in the conversation. This continues until a contact with no reply-to ID is reached or until the replies limit set in the API call is reached.

To stay within Twitter's rate limits, Twitter conversations for tweets are constructed using the reply-to IDs stored in SocialMiner's data store.

Reply-to IDs are stored only for contacts that are retrieved using a Twitter account or Twitter stream feed. (The reply-to ID field is only populated for contacts retrieved through a Twitter account or Twitter stream feed.)

Specifying an older tweet in the conversation will not show the newer tweets in the conversation. The conversation API lists only the replies to the selected contact. It does not list any replies that the selected contact was in-reply-to.



### Note

Twitter direct messages (DMs) are not linked to each other by Twitter, so the conversation aspect of DMs cannot be retrieved.

- [Conversation API Commands, page 51](#)

## Conversation API Commands

This section describes the supported command (GET) for the Twitter conversations API and the parameters for that command.

### GET (List)

Run this API to list the Twitter conversations for tweets (using the reply-to IDs stored in the SocialMiner data store).

|      |   |
|------|---|
| URL: | <code>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/conversation/&lt;scID&gt;/?replies=xxx</code> |
|------|---|

|                        |  |
|------------------------|--|
| <b>HTTP method:</b>    | GET  |
| <b>URL parameters:</b> | <ul style="list-style-type: none"><li>• <b>scID</b> Required, string. The ID of the social contact whose conversation will be retrieved.</li><li>• <b>replies</b> Integer. The maximum number of replies to retrieve. Default is 32.</li></ul> |



|                               |   |
|-------------------------------|---|
| <b>Example http request:</b>  | https://<ServerIP>:<Port>/ccp-webapp/ccp/conversation/25FD59151000012E001FFC6B0A568DF5?replies=3  |
| <b>Example XML response:</b>  | <pre> &lt;SocialContacts&gt;   &lt;SocialContact&gt;     &lt;author&gt;ContactAuthor&lt;/author&gt;     &lt;description&gt;Text of the tweet&lt;/description&gt;     &lt;id&gt;F1217D711000012D000003AB0A568DF5&lt;/id&gt;     &lt;link&gt;       http://twitter.com/ContactAuthor/statuses/33460956554076160     &lt;/link&gt;     &lt;publishedDate&gt;1296827212000&lt;/publishedDate&gt;     &lt;refURL&gt;       http://[ServerIP]:[port]/ccp-webapp/ccp/socialcontact/F1217D711000012D000003AB0A568DF5     &lt;/refURL&gt;     &lt;status&gt;unread&lt;/status&gt;     &lt;statusTimestamp&gt;1296830659988&lt;/statusTimestamp&gt;     &lt;statusUserId/&gt;     &lt;tags/&gt;     &lt;title&gt;Tweet: ContactAuthor to Agent&lt;/title&gt;   &lt;/SocialContact&gt;   &lt;SocialContact&gt;     &lt;author&gt;Agent&lt;/author&gt;     &lt;description&gt;Text of the tweet&lt;/description&gt;     &lt;id&gt;F1217D711000012D000003AB0A568DF5&lt;/id&gt;     &lt;link&gt;       http://twitter.com/ContactAuthor/statuses/33460956554076160     &lt;/link&gt;     &lt;publishedDate&gt;1296827212000&lt;/publishedDate&gt;     &lt;refURL&gt;       http://[ServerIP]:[port]/ccp-webapp/ccp/socialcontact/F1217D711000012D000003AB0A568DF5     &lt;/refURL&gt;     &lt;status&gt;unread&lt;/status&gt;     &lt;statusTimestamp&gt;1296830659988&lt;/statusTimestamp&gt;     &lt;statusUserId/&gt;     &lt;tags/&gt;     &lt;title&gt;Tweet: Agent to ContactAuthor&lt;/title&gt;   &lt;/SocialContact&gt;   &lt;SocialContact&gt;     &lt;author&gt;ContactAuthor&lt;/author&gt;     &lt;description&gt;Text of the tweet&lt;/description&gt;     &lt;id&gt;F1217D711000012D000003AB0A568DF5&lt;/id&gt;     &lt;link&gt;       http://twitter.com/ContactAuthor/statuses/33460956554076160     &lt;/link&gt;     &lt;publishedDate&gt;1296827212000&lt;/publishedDate&gt;     &lt;refURL&gt;       http://[ServerIP]:[port]/ccp-webapp/ccp/socialcontact/F1217D711000012D000003AB0A568DF5     &lt;/refURL&gt;     &lt;status&gt;handled&lt;/status&gt;     &lt;statusTimestamp&gt;1296830659988&lt;/statusTimestamp&gt;     &lt;statusUserId&gt;Agent&lt;/statusUserId&gt;     &lt;tags/&gt;     &lt;title&gt;Tweet: ContactAuthor&lt;/title&gt;   &lt;/SocialContact&gt; &lt;/SocialContacts&gt; </pre> |
| <b>HTTP response headers:</b> | A 200 OK http header is returned on success.  |





# Email

Use the Email API to retrieve the existing SMTP server configuration and to update it if necessary. An SMTP server connection is required to send email notifications.

This API is represented on the SocialMiner user interface in the System Administration panel.

  
**Note**

Only the administrator created during install can use this API.

- [Email API Commands, page 55](#)

## Email API Commands

This section describes the supported commands for Email API and the parameters for those commands.

### Related Topics

- [GET, on page 55](#)
- [PUT, on page 56](#)
- [Email API Parameters, on page 56](#)

## GET

Retrieves the SMTP configuration.

|              |   |
|--------------|---|
| URL:         | http://<ServerIP>:<Port>/ccp-webapp/ccp/email/default |
| HTTP method: | GET   |

|                              |  |
|------------------------------|--|
| <b>Example XML response:</b> | <pre> &lt;Email&gt;   &lt;smtpHost&gt;&lt;ServerIP&gt;:&lt;Port&gt;&lt;/smtpHost&gt;   &lt;smtpPort&gt;587&lt;/smtpPort&gt;   &lt;smtpFromUser&gt;     FromUser@Here.net   &lt;/smtpFromUser&gt;   &lt;smtpHostUserName&gt;     userNameForEmailServer   &lt;/smtpHostUserName&gt;   &lt;smtpAuthenticationEnabled&gt;     true   &lt;/smtpAuthenticationEnabled&gt;   &lt;smtpEnabled&gt;true&lt;/smtpEnabled&gt;   &lt;smtpSslEnabled&gt;true&lt;/smtpSslEnabled&gt;   &lt;refURL&gt;     http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/     email/default   &lt;/refURL&gt; &lt;/Email&gt; </pre> |
| <b>Parameters:</b>           | See <a href="#">Email API Parameters</a> , on page 56.   |

## PUT

Updates the SMTP configuration.



### Note

Only the administrator created during install can use this API.

|                                     |   |
|-------------------------------------|---|
| <b>URL:</b>                         | http://<ServerIP>:<Port>/ccp-webapp/ccp/email/default   |
| <b>HTTP method:</b>                 | PUT   |
| <b>Example XML request payload:</b> | <pre> &lt;Email&gt;   &lt;smtpHost&gt;[ServerIP]&lt;/smtpHost&gt;   &lt;smtpPort&gt;587&lt;/smtpPort&gt;   &lt;smtpFromUser&gt;FromUser@Here.net&lt;/smtpFromUser&gt;   &lt;smtpHostUserName&gt;userNameForEmailServer&lt;/smtpHostUserName&gt;    &lt;smtpHostUserPassword&gt;userPasswordForEmailServer&lt;/smtpHostUserPassword&gt;    &lt;smtpAuthenticationEnabled&gt;true&lt;/smtpAuthenticationEnabled&gt;   &lt;smtpEnabled&gt;true&lt;/smtpEnabled&gt;   &lt;refURL&gt;http://[ServerIP]/ccp-webapp/ccp/email/default&lt;/refURL&gt; &lt;/Email&gt; </pre> |
| <b>Parameters:</b>                  | See <a href="#">Email API Parameters</a> , on page 56.  |

## Email API Parameters

Parameters are optional unless otherwise noted.

| Parameter          | Description                                 | Notes                       |
|--------------------|---|-----------------------------|
| <b>smtpEnabled</b> | Whether this SMTP configuration is enabled. | Boolean. Defaults to false. |

| Parameter                        | Description  | Notes   |
|----------------------------------|--|---|
| <b>smtpAuthenticationEnabled</b> | Whether SMTP authentication is required for the SMTP host. | Boolean. Defaults to false.                     |
| <b>smtpFromUser</b>              | The email (reply-to) address of email sent by this server. | Required when smtpEnabled = true.               |
| <b>smtpHost</b>                  | The fully qualified host address of the SMTP server.       | Required when smtpEnabled = true.               |
| <b>smtpHostUserName</b>          | The username used to log into the SMTP server.             | Required when smtpAuthenticationEnabled = true. |
| <b>smtpHostUserPassword</b>      | The password used to log into the SMTP server.             | Required when smtpAuthenticationEnabled = true. |
| <b>smtpPort</b>                  | The SMTP port. Default is 587.                             | Required when smtpEnabled = true.               |
| <b>smtpSslEnabled</b>            | Whether SSL is enabled for the SMTP server.                | Boolean. Defaults to true.                      |





## Facebook Reply

The Facebook reply API works much like [Twitter Reply](#), on page 171.

See also [Facebook Account Authorization](#), on page 79.

- [Facebook Reply API Commands](#), page 59

## Facebook Reply API Commands

This section describes the supported commands for the Facebook reply API and the parameters for those commands.

### Related Topics

- [GET](#), on page 59
- [POST \(Comment\)](#), on page 60
- [POST \(Like\)](#), on page 61
- [GET \(Like\)](#), on page 62
- [GET \(User\)](#), on page 62

## GET

Gets the status of a Facebook reply API call.

|                              |   |
|------------------------------|---|
| <b>URL:</b>                  | <code>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/reply/facebook/&lt;id&gt;</code><br>For more information about the elements in the URL, see <a href="#">API Conventions</a> , on page 1.<br>In this instance, <id> represents the ProgressID being requested. |
| <b>HTTP method:</b>          | GET   |
| <b>Example XML response:</b> | If the fields in the request are valid, the Location field in the response contains the URL for the social contact associated with the post being queried.  |

|                          |   |
|--------------------------|---|
|                          | <p>If the operation succeeds, the response returns the following XML:</p> <pre> &lt;RequestProgress&gt;   &lt;apiErrors&gt;     &lt;apiError&gt;       &lt;ErrorType&gt;&lt;/ErrorType&gt;       &lt;ErrorData&gt;&lt;/ErrorData&gt;       &lt;ErrorMessage&gt;&lt;/ErrorMessage&gt;     &lt;/apiError&gt;   &lt;/apiErrors&gt;   &lt;httpResponseCode&gt;responseCode&lt;/httpResponseCode&gt;   &lt;httpResponseMessage&gt;responseMessage&lt;/httpResponseMessage&gt;   &lt;progress&gt;SUCCEEDED   FAILED   IN-PROGRESS&lt;/progress&gt;   &lt;Facebook&gt;     &lt;postId&gt;facebookPostId&lt;/PostId&gt;   &lt;/Facebook&gt; &lt;/RequestProgress&gt; </pre> <p>If the operation fails, the <code>httpResponseCode</code> and <code>httpResponseMessage</code> fields contain the code and message returned by Facebook. The <code>apiErrors</code> field can contain additional detailed information about the error.</p>                           |
| <b>Response payload:</b> | <ul style="list-style-type: none"> <li>• <b>httpResponseCode:</b> the response code received from Facebook.</li> <li>• <b>httpResponseMessage:</b> the response message received from Facebook.</li> <li>• <b>progress:</b> <ul style="list-style-type: none"> <li>◦ IN_PROGRESS if SocialMiner is waiting for a response from Facebook.</li> <li>◦ SUCCEEDED if the Facebook operation succeeded.</li> <li>◦ FAILED if the Facebook operation failed. Use <code>httpResponseCode</code> and <code>httpResponseMessage</code> to determine why the operation failed.</li> </ul> </li> <li>• <b>facebookReplyPostId:</b> the Facebook Post ID of the comment.</li> <li>• <b>facebookErrorType:</b> the Facebook error type returned from Facebook if the <code>httpResponseCode</code> is NOT_FOUND.</li> <li>• <b>facebookErrorMessage:</b> the Facebook error message returned from Facebook if the <code>httpResponseCode</code> is NOT_FOUND.</li> </ul> |

## POST (Comment)

Sends a status message (reply or comment) for a Facebook post.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | <p><code>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/reply/facebook/comment</code></p> <p>For more information about the elements in the URL, see <a href="#">API Conventions</a>, on page 1.</p>             |
| <b>HTTP method:</b> | POST  |
| <b>Parameters:</b>  | <ul style="list-style-type: none"> <li>• <b>socialContact:</b> Is the RefURL of the social contact for the Facebook post (each Facebook post is represented by a different social contact in SocialMiner).</li> </ul> |



|                                     |   |
|-------------------------------------|---|
|                                     | <ul style="list-style-type: none"> <li>• <b>message</b>: The text of the status message.</li> </ul>   |
| <b>Example XML request payload:</b> | <pre>&lt;Comment&gt;   &lt;socialContact&gt;socialContactRefUrl&lt;/socialContact&gt;   &lt;message&gt;status text&lt;/message&gt; &lt;/Comment&gt;</pre>   |
| <b>Response:</b>                    | <p>If the fields in the request are valid, the Location field in the response contains the URL for the social contact associated with the post.</p> <p>If the operation succeeds, the response returns the following XML:</p> <pre>&lt;RequestProgress&gt;   &lt;apiErrors&gt;     &lt;apiError&gt;       &lt;ErrorType&gt;&lt;/ErrorType&gt;       &lt;ErrorData&gt;&lt;/ErrorData&gt;       &lt;ErrorMessage&gt;&lt;/ErrorMessage&gt;     &lt;/apiError&gt;   &lt;/apiErrors&gt;   &lt;progress&gt;SUCCEEDED&lt;/progress&gt; &lt;/RequestProgress&gt;</pre> <p>If the operation fails, the httpResponseCode and httpResponseMessage fields contain the code and message returned by Facebook. The apiErrors field can contain additional detailed information about the error.</p> |

## POST (Like)

Use this to 'like' a specific post on Facebook.

|                                     |   |
|-------------------------------------|---|
| <b>URL:</b>                         | http://<ServerIP>:<Port>/ccp-webapp/ccp/reply/facebook/like   |
| <b>HTTP method:</b>                 | POST  |
| <b>Parameters:</b>                  | <ul style="list-style-type: none"> <li>• <b>socialContact</b>: Is the RefURL of the social contact for the Facebook post being 'liked'.</li> <li>• <b>likes</b>: Like or Unlike a post. Boolean.</li> </ul>   |
| <b>Example XML request payload:</b> | <pre>&lt;Like&gt;   &lt;socialContact&gt;socialContactRefUrl&lt;/socialContact&gt;   &lt;likes&gt;booleanValue&lt;/likes&gt; &lt;/Like&gt;</pre>  |
| <b>Response:</b>                    | <p>If the fields in the request are valid, the Location field in the response contains the URL for the social contact associated with the post being 'liked'.</p> <p>If the operation succeeds, the response returns the following XML:</p> <pre>&lt;RequestProgress&gt;   &lt;apiErrors&gt;     &lt;apiError&gt;       &lt;ErrorType&gt;&lt;/ErrorType&gt;       &lt;ErrorData&gt;&lt;/ErrorData&gt;       &lt;ErrorMessage&gt;&lt;/ErrorMessage&gt;</pre> |

```

</apiError>
</apiErrors>
<progress>SUCCEEDED</progress>
</RequestProgress>

```

If the operation fails, the `httpResponseCode` and `httpResponseMessage` fields contain the code and message returned by Facebook. The `apiErrors` field can contain additional detailed information about the error.

## GET (Like)

Use this to determine if the authorized user already likes a post on Facebook.

|                     |  |
|---------------------|--|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/reply/facebook/like  |
| <b>HTTP method:</b> | GET  |
| <b>Parameters:</b>  | <b>socialContact:</b> the RefURL of the social contact being queried.  |
| <b>Response:</b>    | <p>If the fields in the request are valid, the Location field in the response contains the URL for the social contact.</p> <p>If the operation succeeds, the response returns the following XML:</p> <pre> &lt;RequestProgress&gt;   &lt;apiErrors&gt;     &lt;apiError&gt;       &lt;ErrorType&gt;&lt;/ErrorType&gt;       &lt;ErrorData&gt;&lt;/ErrorData&gt;       &lt;ErrorMessage&gt;&lt;/ErrorMessage&gt;     &lt;/apiError&gt;   &lt;/apiErrors&gt;   &lt;progress&gt;SUCCEEDED FAILED IN-PROGRESS&lt;/progress&gt;   &lt;Facebook&gt;     &lt;like&gt;value&lt;/Like&gt;   &lt;/Facebook&gt; &lt;/RequestProgress&gt; </pre> <p><b>Note</b> <code>like</code> will contain a boolean value of <i>true</i> if the user already likes the post and <i>false</i> if the user does not already like the post .</p> <p>If the operation fails, the <code>httpResponseCode</code> and <code>httpResponseMessage</code> fields contain the code and message returned by Facebook. The <code>apiErrors</code> field can contain additional detailed information about the error.</p> |

## GET (User)

Retrieves the profile information of the Facebook user.

|                     |  |
|---------------------|--|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/reply/facebook/user/   |
| <b>HTTP method:</b> | GET  |
| <b>Parameters:</b>  | <b>socialContact :</b> the RefURL of the social contact for the user whose public Facebook profile data will be retrieved. |

|                              |  |
|------------------------------|--|
| <b>Example http request:</b> | <pre>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/reply/facebook/ user?socialContact=https://xx.yy.zzz.ggg/ ccp-webapp/ccp/socialcontact/ C07769381000014000000081D0A568DDD</pre> <p>If the fields in the request are valid, the Location field of the response contains the URL to poll for the status of the operation, for example:</p> <pre>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/reply/facebook/6</pre> <p>Otherwise, SocialMiner will return an error. See <a href="#">HTTP Responses</a> for error response.</p>   |
| <b>Example XML response:</b> | <p>If the operation fails, the httpResponseCode and httpResponseMessage fields contain the code and message returned by Facebook. The apiErrors field can contain additional detailed information about the error.</p> <p>If the operation with the polling URL (http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/reply/facebook/6) succeeds, the response returns the following XML:</p> <pre>&lt;RequestProgress&gt; &lt;apiErrors/&gt; &lt;httpResponseCode&gt;200&lt;/httpResponseCode&gt; &lt;httpResponseMessage&gt;OK&lt;/httpResponseMessage&gt; &lt;progress&gt;SUCCEEDED&lt;/progress&gt; &lt;facebook/&gt;   &lt;facebookUser&gt;     &lt;id&gt;100004025925472&lt;/id&gt;     &lt;name&gt;Soso Cannons&lt;/name&gt;     &lt;username&gt;soso.cannons&lt;/username&gt;     &lt;link&gt;http://www.facebook.com/soso.cannons&lt;/link&gt;     &lt;profileImageUrl&gt;http://profile.ak.fbcdn.net/hprofile-ak-prn2/       274695_100004025925472_1641583011_q.jpg&lt;/profileImageUrl&gt;   &lt;/facebookUser&gt; &lt;/RequestProgress&gt;</pre> |





## Feed

The Feed API allows you to create, delete, update, and list feeds that retrieve contacts. SocialMiner feeds can be RSS feeds, Twitter accounts, Twitter streams, Twitter searches, Facebook fan pages, push feeds, chat feeds, or callback feeds. (Callback feeds support the callback API. See [Callback](#), on page 13 for more information.)

The feed object contains data about the feed—such as the URL of the feed, how often the feed is to be read (the polling interval), whether SocialMiner needs to access the feed through a proxy, and the feed type.

This API is represented on the SocialMiner user interface by the Feeds panel.

Feeds are assigned to at least one [Campaign](#). To acquire contacts, you must create a feed, create a campaign, and add the feed to the campaign.

- [Feed API Commands](#), page 65
- [Authorize Against Twitter Feeds](#), page 75
- [Authorize Facebook Accounts](#), page 78

## Feed API Commands

This section describes the commands supported for the Feed API and the parameters for those commands. Additional information about push feeds is documented in [Push Feed](#).

### POST

Creates a feed to be stored in the database.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/feed          |
| <b>HTTP method:</b> | POST  |
| <b>Parameters:</b>  | See <a href="#">Feed API Parameters</a> , on page 71. |

## DELETE

|                                     |   |
|-------------------------------------|---|
| <b>Example XML request payload:</b> | <pre> &lt;Feed&gt;   &lt;name&gt;test&lt;/name&gt;   &lt;description&gt;this is the description &lt;/description&gt;   &lt;url&gt;http://test.com&lt;/url&gt;   &lt;type&gt;5&lt;/type&gt;   &lt;pollingInterval&gt;60&lt;/pollingInterval&gt;   &lt;minAge&gt;1&lt;/minAge&gt;   &lt;authenticationUsername&gt;User1 &lt;/authenticationUsername&gt;   &lt;authenticationPassword&gt;password1 &lt;/authenticationPassword&gt;   &lt;replytemplateRefURL&gt;     http://[ServerIP]:[Port]/     ccp-webapp/ccp/template/reply/105678   &lt;/replytemplateRefURL&gt;   &lt;tags&gt;     &lt;tag&gt;tag1&lt;/tag&gt;     &lt;tag&gt;tag2&lt;/tag&gt;   &lt;/tags&gt; &lt;/Feed&gt; </pre> |
| <b>HTTP response headers:</b>       | <p>The response contains the <b>URL</b> for the newly created feed. Note the id: <i>100162</i>.</p> <pre> http/1.1 201 Created Location: http://&lt;ServerIP&gt;:&lt;Port&gt; /ccp-webapp/ccp/feed/100162 Content-Type: text/plain Content-Length: 0 Date: Tue, 12 Jan 2010 16:15:04 GMT </pre> <p>See also <a href="#">HTTP Responses</a>.</p>   |

For feeds that require authorization (such as Twitter feeds and Facebook fan page feeds), this API creates a feed pending on OAuth. The webapp initiates an OAuth session with Twitter or Facebook. SocialMiner allows for one feed per Twitter user account; therefore the webapp evaluates whether or not the username has already been used. Once the OAuth successfully completes, the pending feed is finalized and added to the system.

There are several extra steps for creating Twitter account feeds and Facebook fan page feeds. See [Authorize Against Twitter Feeds](#) and [Facebook Account Authorization](#).

## DELETE

Deletes a feed from the database.

For feeds that require authorization (Twitter account and Facebook fan page feeds), this function deletes the specified feed if it has passed OAuth, or cancels the pending authentication if it has not passed OAuth.

|                               |  |
|-------------------------------|--|
| <b>URL:</b>                   | http://<ServerIP>:<Port>/ccp-webapp/ccp/feed/ <a href="#">ID Variables</a> |
| <b>HTTP method:</b>           | DELETE   |
| <b>HTTP response headers:</b> | See <a href="#">HTTP Responses</a> .                                       |

## GET (List)

Retrieves a list of all feeds in the system.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/feed?summary=<true/false> |
| <b>HTTP method:</b> | GET   |
| <b>Parameters:</b>  | See <a href="#">Feed API Parameters</a> , on page 71.             |

|                              |   |
|------------------------------|---|
| <b>Example:</b>              | <code>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/feed?summary=false</code>   |
| <b>Example XML response:</b> | <pre> &lt;feeds&gt;  &lt;Feed&gt; &lt;changeStamp&gt;0&lt;/changeStamp&gt; &lt;name&gt;Boston.com Most Popular&lt;/name&gt; &lt;pollingInterval&gt;300&lt;/pollingInterval&gt;  &lt;refURL&gt; http://[ServerIP]:[Port]/ccp-webapp/ccp/feed/100347 &lt;/refURL&gt; &lt;status&gt;1&lt;/status&gt; &lt;tags/&gt; &lt;type&gt;1&lt;/type&gt; &lt;url&gt;http://feeds.boston.com/boston/mostpopular&lt;/url&gt; &lt;/Feed&gt;  &lt;Feed&gt; &lt;changeStamp&gt;2&lt;/changeStamp&gt; &lt;name&gt;Boston.com top stories&lt;/name&gt; &lt;pollingInterval&gt;300&lt;/pollingInterval&gt;  &lt;refURL&gt; http://[ServerIP]:[Port]/ccp-webapp/ccp/feed/100346 &lt;/refURL&gt; &lt;status&gt;1&lt;/status&gt; &lt;tags/&gt; &lt;type&gt;1&lt;/type&gt; &lt;url&gt;http://feeds.boston.com/boston/topstories&lt;/url&gt; &lt;/Feed&gt;  &lt;Feed&gt; &lt;changeStamp&gt;0&lt;/changeStamp&gt; &lt;name&gt;Cisco Live&lt;/name&gt; &lt;pollingInterval&gt;300&lt;/pollingInterval&gt;  &lt;refURL&gt; http://[ServerIP]:[Port]/ccp-webapp/ccp/feed/102638 &lt;/refURL&gt;  &lt;replyTemplateRefURL&gt; http://[ServerIP]:[Port]/ccp-webapp/ccp/template/reply/301 &lt;/replyTemplateRefURL&gt; &lt;status&gt;1&lt;/status&gt; &lt;tags/&gt; &lt;type&gt;4&lt;/type&gt; &lt;url&gt;http://www.facebook.com/ciscoliveeurope&lt;/url&gt; &lt;/Feed&gt;  &lt;Feed&gt; &lt;changeStamp&gt;0&lt;/changeStamp&gt; &lt;name&gt;contacts&lt;/name&gt;  &lt;pushFeedURL&gt; http://[ServerIP]:[Port]/ccp-webapp/ccp/pushfeed/102619 &lt;/pushFeedURL&gt;  &lt;refURL&gt; http://[ServerIP]:[Port]/ccp-webapp/ccp/feed/102619 &lt;/refURL&gt; &lt;status&gt;1&lt;/status&gt; &lt;tags/&gt; &lt;type&gt;7&lt;/type&gt; &lt;/Feed&gt;  &lt;Feed&gt; &lt;changeStamp&gt;2&lt;/changeStamp&gt; &lt;name&gt;Twitter boston&lt;/name&gt; &lt;pollingInterval&gt;180&lt;/pollingInterval&gt; </pre> |



|                               |   |
|-------------------------------|---|
|                               | <pre> &lt;refURL&gt; http://[ServerIP]:[Port]/ccp-webapp/ccp/feed/102621 &lt;/refURL&gt;  &lt;replyTemplateRefURL&gt; http://[ServerIP]:[Port]/ccp-webapp/ccp/template/reply/103184 &lt;/replyTemplateRefURL&gt; &lt;status&gt;1&lt;/status&gt; &lt;tags/&gt; &lt;type&gt;1&lt;/type&gt; &lt;url&gt;http://search.twitter.com/search.rss?q=boston&lt;/url&gt; &lt;/Feed&gt; &lt;/feeds&gt; </pre> |
| <b>HTTP response headers:</b> | <pre> http/1.1 200 OK Content-Type: application/xml Transfer-Encoding: chunked Date: Tue, 12 Jan 2010 16:47:58 GMT </pre> <p>See also <a href="#">HTTP Responses</a>.</p>   |

## GET

Returns the data for a single feed. For security, passwords are not returned for feeds. Password elements are masked (\*\*\*\*\*).

|                     |   |
|---------------------|---|
| <b>URL:</b>         | <pre>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/feed/&lt;id&gt;</pre> <p>For more information about &lt;id&gt;, see <a href="#">ID Variables</a>, on page 2.</p> |
| <b>HTTP method:</b> | GET   |

|                               |   |
|-------------------------------|---|
| <b>Example XML response:</b>  | <pre> &lt;Feed&gt;   &lt;refURL&gt;     http://[ServerIP]:[Port]/ccp-webapp/ccp/     feed/(id)   &lt;/refURL&gt;   &lt;name&gt;test&lt;/name&gt;   &lt;description&gt;this is the description&lt;/description&gt;   &lt;url&gt;http://test.com&lt;/url&gt;   &lt;type&gt;1&lt;/type&gt;   &lt;pollingInterval&gt;60&lt;/pollingInterval&gt;   &lt;useProxy&gt;false&lt;/useProxy&gt;   &lt;minAge&gt;1&lt;/minAge&gt;   &lt;changeStamp&gt;0&lt;/changeStamp&gt;   &lt;sessionToken&gt;*****&lt;/sessionToken&gt;   &lt;status&gt;0&lt;/status&gt;   &lt;replytemplateRefURL&gt;http://[ServerIP]:[Port]/     ccp-webapp/ccp/template/reply/105678   &lt;/replytemplateRefURL&gt;   &lt;tags&gt;     &lt;tag&gt;tag1&lt;/tag&gt;     &lt;tag&gt;tag2&lt;/tag&gt;   &lt;/tags&gt; &lt;/Feed&gt; </pre> |
| <b>HTTP response headers:</b> | <pre> http/1.1 200 OK Content-Type: application/xml Transfer-Encoding: chunked Date: Tue, 12 Jan 2010 16:50:46 GMT </pre> <p>See also <a href="#">HTTP Responses</a>.</p>   |

## PUT

Updates an existing feed.

For feeds that require authorization (Twitter account and Facebook fan page feeds), this API updates an existing feed which has passed OAuth. If the username is changed, or if the input XML document includes “<status>3</status>”, the create procedure is invoked for re-OAuth. Otherwise, it works in the same way as updating a feed without OAuth.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | <pre>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/feed/&lt;id&gt;</pre> <p>For more information about &lt;id&gt;, see <a href="#">ID Variables</a>, on page 2.</p> |
| <b>HTTP method:</b> | PUT   |
| <b>Parameters:</b>  | See <a href="#">Feed API Parameters</a> , on page 71.   |

|                                     |  |
|-------------------------------------|--|
| <b>Example XML request payload:</b> | <pre> &lt;Feed&gt;   &lt;name&gt;this is the new name&lt;/name&gt;   &lt;description&gt;     this is an updated description   &lt;/description&gt;   &lt;url&gt;http://test.com&lt;/url&gt;   &lt;type&gt;1&lt;/type&gt;   &lt;pollingInterval&gt;60&lt;/pollingInterval&gt;   &lt;useProxy&gt;&gt;false&lt;/useProxy&gt;   &lt;minAge&gt;1&lt;/minAge&gt;   &lt;changeStamp&gt;0&lt;/changeStamp&gt;   &lt;authToken&gt;WJFH837923&lt;/authToken&gt;   &lt;status&gt;0&lt;/status&gt;   &lt;replytemplateRefURL&gt;     http://[ServerIP]:[Port]/     ccp-webapp/ccp/template/reply/105678   &lt;/replytemplateRefURL&gt;   &lt;tags&gt;     &lt;tag&gt;tag1&lt;/tag&gt;     &lt;tag&gt;tag2&lt;/tag&gt;   &lt;/tags&gt; &lt;/Feed&gt; </pre> |
| <b>HTTP response headers:</b>       | <pre> http/1.1 200 OK Content-Type: text/plain Content-Length: 0 Date: Thu, 14 Jan 2010 15:49:17 GMT </pre> <p>See also <a href="#">HTTP Responses</a>.</p>  |

## Feed API Parameters

This table defines the parameters used by the feed API. The table below this one identifies which parameters apply to each feed type and whether they are required or optional.

| Parameter name                | Description   | Notes  |
|-------------------------------|---|--|
| <b>authenticationPassword</b> | The password for the username provided for an account.                    | String   |
| <b>authenticationUsername</b> | The username for an account.  | String   |
| <b>authToken</b>              | This is the OAuth access token.   | String   |
| <b>changeStamp</b>            | The change stamp of the feed record.                                      | Integer. Defaults to 0.<br>Required for PUT (update).<br>Is returned in GET. |
| <b>chatJoinTimeout</b>        | The amount of time (in seconds) that the agent has to join the chat room. |  |

| Parameter name               | Description   | Notes   |
|------------------------------|---|---|
| <b>chatInactivityTimeout</b> | The amount of idle time (in seconds) between chat messages. If a chat message is not sent in this amount of time, the chat session is taken down. |   |
| <b>description</b>           | The description of the feed.  | String  |
| <b>keywords</b>              | The search criteria required by TWITTER_SEARCH, or the comma separated list of keywords to search for with the TWITTER_STREAM.                    | <p>Restrictions apply but based on the restrictions by Twitter on their search or stream api respectively. There must be a least one keyword defined.</p> <p>For TWITTER_STREAM, up to 200 keywords can be defined for a total limit of 2000 bytes. Each keyword must be between 1-60 bytes.</p> <p>For TWITTER_SEARCH, up to 1000 characters are allowed to form the search query, including operators.</p> <p>Spaces are interpreted as an “and” modifier for search.</p> |
| <b>minAge</b>                | The minimum post age in seconds (defaults to 0).  | If a post is newer than the <i>minAge</i> , it will not be stored by the feed.  |
| <b>name</b>                  | The name of the feed.   | String, must be unique.<br>Required for creating (POST).  |
| <b>pollingInterval</b>       | The amount of time in seconds the system waits between attempts to read this feed.  | Integer   |
| <b>pushFeedURL</b>           | The URL to which you push the entities that will become social contacts.  |   |
| <b>refURL</b>                | A copy of the URL requested.  | Response for GET.   |
| <b>replytemplateRefURL</b>   | The URL of the reply template used to respond to social contacts obtained from this feed.   | <p>String</p> <p>If this field is blank, no reply template is used.</p>   |

| Parameter name | Description   | Notes  |
|----------------|---|--|
| <b>status</b>  | The authorization status for Facebook, Twitter search, and Twitter account.             | Values are: <ul style="list-style-type: none"> <li>• AUTHENTICATION_SUCCEEDED=0</li> <li>• AUTHENTICATION_NONE=1</li> <li>• AUTHENTICATION_FAILED=2</li> <li>• AUTHENTICATION_PENDING = 3</li> </ul>   |
| <b>summary</b> | Determines whether full object information or URLs only are returned for the list.      | Boolean. Defaults to false. When “true”, only the URLs of the objects are returned. If summary=false, full object information, along with the URL reference, is returned. URL Parameter. Used for List API only.   |
| <b>tags</b>    | Contacts coming in from this feed will automatically be tagged with these default tags. | String. A maximum of 10 tags are allowed.  |
| <b>type</b>    | The feed type.  | Required for POST<br>Integer, types are: <ul style="list-style-type: none"> <li>• RSS = 1</li> <li>• Twitter stream = 3</li> <li>• Facebook fan page = 4</li> <li>• Authenticated RSS = 5</li> <li>• Twitter account = 6</li> <li>• Push = 7</li> <li>• Chat = 8</li> <li>• Twitter search = 9</li> <li>• Callback = 10</li> </ul> |
| <b>url</b>     | The location of the feed that you want to read.   | String<br>If type is FACEBOOK, you cannot use an IP address; you must use a hostname.  |

This table summarizes the fields used by each feed type as R (required), O (optional), or NA (not applicable).

| <b>Feed type/<br/>Field name</b>       | <b>RSS<br/>(1)</b> | <b>Twitter<br/>stream<br/>(3)</b> | <b>Facebook<br/>(4)</b> | <b>Auth RSS<br/>(5)</b> | <b>Twitter<br/>account<br/>(6)</b> | <b>Push<br/>(7)</b> | <b>Chat<br/>(8)</b> | <b>Twitter<br/>search (9)</b> | <b>Callback<br/>(10)</b> |
|--|--------------------|-----------------------------------|-------------------------|-------------------------|------------------------------------|---------------------|---------------------|-------------------------------|--------------------------|
| <b>type</b>                            | <b>R</b>           | <b>R</b>                          | <b>R</b>                | <b>R</b>                | <b>R</b>                           | <b>R</b>            | <b>R</b>            | <b>R</b>                      | <b>R</b>                 |
| <b>name</b>                            | <b>R</b>           | <b>R</b>                          | <b>R</b>                | <b>R</b>                | <b>R</b>                           | <b>R</b>            | <b>R</b>            | <b>R</b>                      | <b>R</b>                 |
| <b>description</b>                     | <b>O</b>           | <b>O</b>                          | <b>O</b>                | <b>O</b>                | <b>O</b>                           | <b>O</b>            | <b>O</b>            | <b>O</b>                      | <b>O</b>                 |
| <b>url</b>                             | <b>R</b>           | NA                                | <b>R</b>                | <b>R</b>                | NA                                 | NA                  | NA                  | NA                            | NA                       |
| <b>polling<br/>interval</b>            | <b>R</b>           | NA                                | <b>R</b>                | <b>R</b>                | <b>R</b>                           | NA                  | NA                  | <b>R</b>                      | NA                       |
| <b>minAge</b>                          | <b>O</b>           | NA                                | <b>O</b>                | <b>O</b>                | <b>O</b>                           | NA                  | NA                  | NA                            | NA                       |
| <b>authentication<br/>Username</b>     | NA                 | <b>R</b>                          | NA                      | <b>R</b>                | <b>R</b>                           | NA                  | NA                  | <b>R</b>                      | NA                       |
| <b>authentication<br/>Password</b>     | NA                 | NA                                | NA                      | <b>O</b>                | NA                                 | NA                  | NA                  | NA                            | NA                       |
| <b>keywords</b>                        | NA                 | <b>R</b>                          | NA                      | NA                      | NA                                 | NA                  | NA                  | <b>R</b>                      | NA                       |
| <b>authToken</b>                       | NA                 | NA                                | NA                      | NA                      | NA                                 | NA                  | NA                  | NA                            | NA                       |
| <b>chatJoin<br/>Timeout</b>            | NA                 | NA                                | NA                      | NA                      | NA                                 | NA                  | <b>O</b>            | NA                            | NA                       |
| <b>chat<br/>Inactivity<br/>Timeout</b> | NA                 | NA                                | NA                      | NA                      | NA                                 | NA                  | <b>O</b>            | NA                            | NA                       |
| <b>reply<br/>Template<br/>RefUrl</b>   | <b>O</b>           | <b>O</b>                          | <b>O</b>                | <b>O</b>                | <b>O</b>                           | <b>O</b>            | <b>O</b>            | <b>O</b>                      | <b>O</b>                 |
| <b>tags</b>                            | <b>O</b>           | <b>O</b>                          | <b>O</b>                | <b>O</b>                | <b>O</b>                           | <b>O</b>            | <b>O</b>            | <b>O</b>                      | <b>O</b>                 |

## Authorize Against Twitter Feeds

Twitter uses [OAuth](#) to provide secure communication between Twitter and third-party applications such as SocialMiner. You must obtain authorization to access a Twitter account for any kind of Twitter feed type (account, stream, or search).

For this reason, there are several additional steps to creating a Twitter feed:

### Procedure

- 
- Step 1** Create a Twitter feed.  
See [POST](#), on page 65 for more information about how to do this.
  - Step 2** Use [oauthGetStatus](#), on page 76 to poll the feed status until the *status* is WAITING-FOR-AUTH-CALLBACK. The *authUrl* field is populated with the twitter authorization URL.
  - Step 3** Access the **authUrl** through a browser session and allow authorization for SocialMiner. Twitter returns a PIN code through [oauthCallback](#), on page 77. This is handled on the server and does not require any API calls from your application.
  - Step 4** Poll with [oauthGetStatus](#), on page 76 until the status is *SUCCEEDED*. The feed has been created.
- 



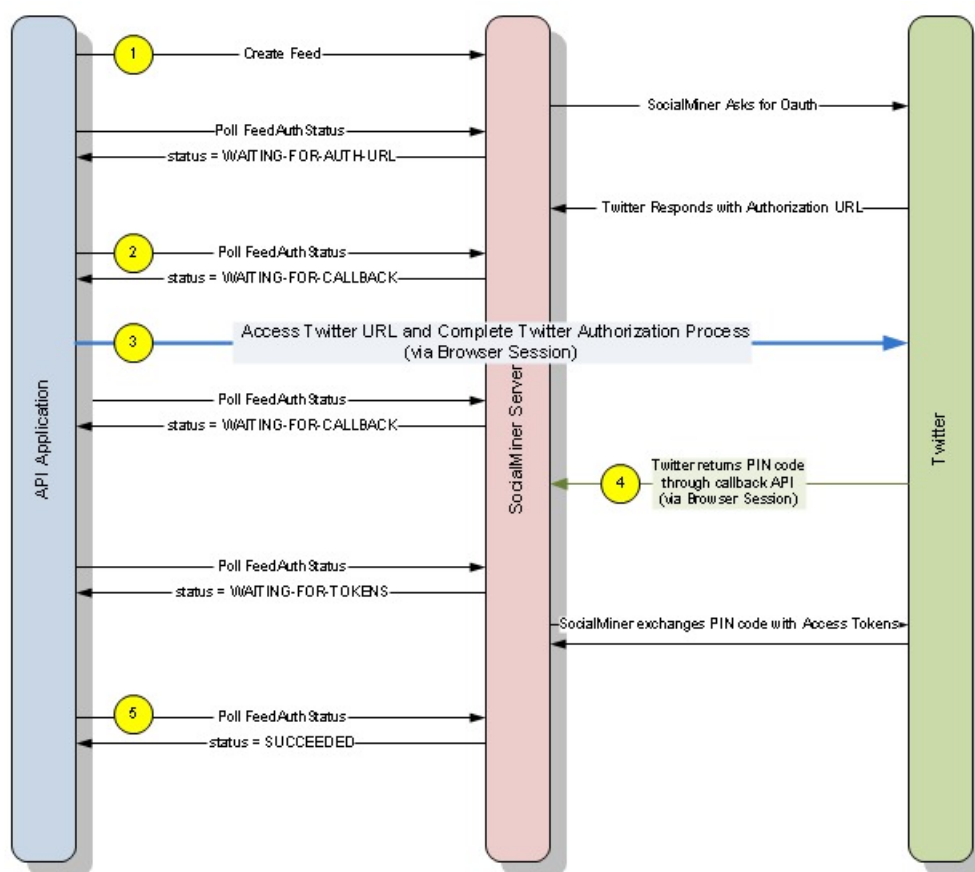
#### Note

The Shindig OpenSocial container in which SocialMiner runs requires that REST requests complete within five seconds. Communication with the Twitter servers can exceed five seconds. This limitation means that after making calls to the Feed API for creating Twitter feeds, you must poll to verify the status returned.

## Twitter Account Authorization Communication

This diagram illustrates the API calls and expected poll responses for Twitter account authorization.

**Figure 1: Twitter Account Authorization Communication Diagram**



## Twitter authorization APIs

### oauthGetStatus

Used for Twitter account, search, and stream feeds. Checks the status of the authorization with Twitter. Your application should poll this URL until status is "SUCCEEDED."

The webapp first checks the hashmap using the input username as a key to see if the feed is there. If it is, the authorization is still pending; otherwise, it checks the database. If the feed is found in the database, the authorization is done successfully. The authorization fails if the feed is not found anywhere.

If the authenticating user is different from the configured user for the feed, the user is allowed to log into Twitter and click "Allow", but SocialMiner will fail the authentication and set the status code to "FAILED-USER-MISMATCH."

|                     |   |
|---------------------|---|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/feed/oauth/<username> |
| <b>HTTP method:</b> | GET   |



**Example XML response:**

```
<FeedOAuthStatus>
  <status>STATUS</status>
</FeedOAuthStatus>
```

- **status** is one of:
  - **SUCCEEDED**—The application/user account has been authorized for use with twitter
  - **FAILED**—The application/user is not authorized for use with twitter. This could be due to a failed login or a timeout. The authorization request times out in 10 minutes.
  - **FAILED-USER-MISMATCH**—The username entered into the twitter authorization page does not match the username on the feed.
  - **IN-PROGRESS**—The authorization is in progress.

**oauthCancel**

Used for Twitter account, search, and stream feeds. Cancels a pending OAuth request.

**Note**

If the OAuth request to Twitter is completed before the call to `oauthCancel` is made, the configuration is not deleted.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | <code>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/feed/oauth/&lt;username&gt;</code>  |
| <b>HTTP method:</b> | DELETE  |
| <b>Response:</b>    | Response is contained in the http response code.<br>See also <a href="#">HTTP Responses</a> . |

**oauthCallback**

Used for Twitter account, search, and stream feeds. This URL is the callback URL invoked by Twitter after a user approves or denies the authorization request. Details are provided here only for reference. Do not call this API in your application.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | <code>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/feed/oauth/&lt;username&gt;/callback</code> |
| <b>HTTP method:</b> | GET   |
| <b>Responses:</b>   | <i>SUCCEEDED</i> or <i>Failed</i> .   |

# Authorize Facebook Accounts

Facebook uses [OAuth](#) to provide secure communication and is similar to Twitter account authorization.

For Facebook fan page feeds, this API creates the feed and starts the OAuth process. The feed is saved in a hashmap indexed by the lookup key. The webapp then creates a future task which initiates an OAuth session with Facebook. This generates an authentication link and saves it with the original hashed feed configuration. Upon success, the function returns the URL for checking the OAuth status and sets the http response code to 202 (Accepted).

There are several additional steps to creating a Facebook fan page feed:

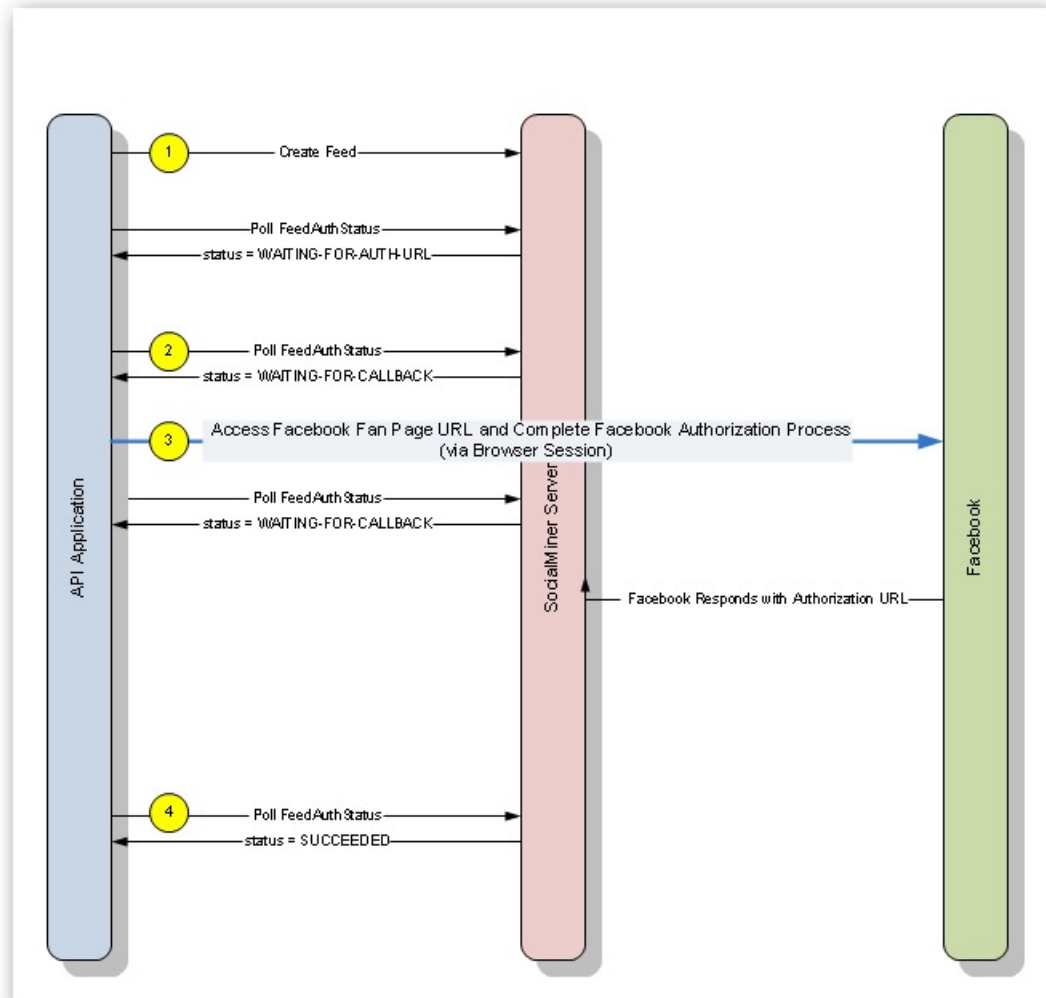
## Procedure

- 
- Step 1** Create a type 4 feed.  
See [POST](#), on page 65 for more information about how to do this.
  - Step 2** Use [FacebookOAuthGetStatus](#), on page 79 to poll the feed status until the status is WAITING-FOR-AUTH-CALLBACK. The authUrl field is populated with the Facebook authorization URL.
  - Step 3** Access the **authUrl** through a browser session and allow authorization for SocialMiner. Facebook returns a PIN code through [FacebookCallback](#), on page 80. This is handled on the server and does not require any API calls from your application.
  - Step 4** Poll with [FacebookOAuthGetStatus](#), on page 79 until the status is "SUCCEEDED". The feed has been created.
-

## Facebook Account Authorization

This diagram illustrates the API calls and expected poll responses for Facebook account authorization.

**Figure 2: Facebook Account Authorization Diagram**



## Facebook authorization APIs

### FacebookOauthGetStatus

Used only for the Facebook fan page feed type. Checks the status of the authorization with Facebook. Your application should poll this URL until status is "SUCCEEDED".

The webapp first checks the hashmap using the input lookup key to see if the feed is there. If it is, the authorization is still pending; otherwise, it checks the database. If the feed is found in the database, the authorization is successful. The authorization fails if the feed is not found anywhere.

If the lookup key is different from the one configured for the feed, the user is allowed to log into Facebook and click **Allow**, but SocialMiner will fail the authentication and set the status code to FAILED-LOOKUP-KEY-MISMATCH.

|                              |  |
|------------------------------|--|
| <b>URL:</b>                  | http://<ServerIP>:<Port>/ccp-webapp/ccp/feed/<lookup-key>/facebookOauth  |
| <b>HTTP method:</b>          | GET  |
| <b>Example XML response:</b> | <pre>&lt;FeedOAuthStatus&gt;   &lt;authUrl&gt;AUTH_URL&lt;/authUrl&gt;   &lt;status&gt;STATUS&lt;/status&gt; &lt;/FeedOAuthStatus&gt;</pre> <ul style="list-style-type: none"> <li>• <b>authURL</b> is the authorization URL from Facebook.</li> <li>• <b>status</b> is one of:           <ul style="list-style-type: none"> <li>◦ <b>SUCCEEDED</b>—The application/user account has been authorized for use with Facebook</li> <li>◦ <b>FAILED</b>—The application/user is not authorized for use with Facebook. This could be due to a failed login or a timeout. The authorization request times out in 10 minutes.</li> <li>◦ <b>FAILED-LOOKUP-KEY-MISMATCH</b>—The lookup key provided does not match the one for the user logged into Facebook.</li> <li>◦ <b>IN-PROGRESS</b>—The authorization is in progress.</li> </ul> </li> </ul> |

## FacebookOAuthCancel

This API cancels a pending OAuth and sets the status to failed. If the OAuth has already completed successfully when the cancel request is received, the saved configuration will not be deleted.

|                         |   |
|-------------------------|---|
| <b>URL:</b>             | http://<ServerIP>:<Port>/ccp-webapp/ccp/feed/<id>/facebookOAuthCancel<br>For more information about <id>, see <a href="#">ID Variables</a> , on page 2. |
| <b>HTTP method:</b>     | DELETE  |
| <b>Request payload:</b> | None  |
| <b>Responses:</b>       | See <a href="#">HTTP Responses</a> , on page 4.   |

## FacebookCallback

Used only for the Facebook account feed, this URL is the callback URL invoked by Facebook after a user approves or denies the authorization request. Details are provided here only for reference. Do not call this API in your application.

|                         |  |
|-------------------------|--|
| <b>URL:</b>             | http://<ServerIP>:<Port>/ccp-webapp/ccp/feed/<id>/facebookCallback<br>For more information about <id>, see <a href="#">ID Variables</a> , on page 2. |
| <b>HTTP method:</b>     | POST   |
| <b>Request payload:</b> | The Facebook-assigned authorization PIN code carried in the “access_code” parameter or the error in parameter “error_description”.                   |
| <b>Responses:</b>       | “Succeeded” or “Failed”.   |





## Filter

The Filter API allows you create, update, and delete filters.

This API is represented on the SocialMiner user interface in the Filters panel.

- [Filter API Commands](#), page 83
- [About Script Filters](#), page 88

## Filter API Commands

This section describes the supported commands for filter API and the parameters for those commands.

Note that for the POST and PUT request payloads for script filters (type = 4), you must enclose the script content in a [CDATA wrapper](#). Failure to do this results in parser errors.

### Related Topics

- [POST](#), on page 83
- [DELETE](#), on page 84
- [GET \(List\)](#), on page 84
- [GET](#), on page 85
- [GET \(Script Content\)](#), on page 86
- [PUT](#), on page 86
- [Filter API Parameters](#), on page 87

## POST

Creates a filter to be stored in the database.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | <code>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/filter</code> |
| <b>HTTP method:</b> | POST  |
| <b>Parameters:</b>  | See <a href="#">Filter API Parameters</a> , on page 87.                 |

## DELETE

|  |  |
|--|--|
| <b>Example XML request payload for Bayesian filter:</b>                          | <pre>&lt;Filter&gt;   &lt;name&gt;Bayesian&lt;/name&gt;   &lt;description&gt;Bayesian&lt;/description&gt;   &lt;type&gt;2&lt;/type&gt; &lt;/Filter&gt;</pre>   |
| <b>Example XML request payload for author filter:</b>                            | <pre>&lt;Filter&gt;   &lt;name&gt;String&lt;/name&gt;   &lt;description&gt;String&lt;/description&gt;   &lt;type&gt;3&lt;/type&gt;   &lt;keywords&gt;     &lt;keyword&gt;author name&lt;/keyword&gt;     &lt;keyword&gt;author userid&lt;/keyword&gt;   &lt;/keywords&gt;   &lt;rule&gt;1&lt;/rule&gt; &lt;/Filter&gt;</pre>           |
| <b>Example XML request payload for script filter, showing the CDATA wrapper:</b> | <pre>&lt;Filter&gt;   &lt;name&gt;String&lt;/name&gt;   &lt;description&gt;String&lt;/description&gt;   &lt;type&gt;4&lt;/type&gt;   &lt;scriptFileName&gt;me.groovy&lt;/scriptFileName&gt;   &lt;scriptContent&gt;     &lt;![CDATA[       put the text of your filter here.     ]]&gt;   &lt;/scriptContent&gt; &lt;/Filter&gt;</pre> |
| <b>HTTP response headers:</b>  | <p>The response contains the URL for the newly created filter.</p> <pre>http/1.1 201 Created Location: http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/filter/1266345862276 Content-Type: text/plain Content-Length: 0 Date: Tue, 16 Feb 2010 19:35:56 GMT</pre>   |

## DELETE

Removes a filter from the database.

|                               |   |
|-------------------------------|---|
| <b>URL:</b>                   | <a href="http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/filter/&lt;ID Variables&gt;">http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/filter/&lt;ID Variables&gt;</a> |
| <b>HTTP method:</b>           | DELETE  |
| <b>HTTP response headers:</b> | <pre>http/1.1 200 OK Content-Type: text/plain Content-Length: 0 Date: Tue, 12 Jan 2010 17:03:54 GMT</pre>   |

## GET (List)

Returns a list of all filters.

|             |   |
|-------------|---|
| <b>URL:</b> | <a href="http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/filter">http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/filter</a> |
|-------------|---|



|                               |   |
|-------------------------------|---|
| <b>HTTP method:</b>           | GET   |
| <b>URL parameter:</b>         | True or false. Defaults to false.<br><br>When “true”, only the URLs of the objects are returned. When “false”, full object information is returned along with the URL reference.  |
| <b>Parameters:</b>            | See <a href="#">Filter API Parameters</a> , on page 87.   |
| <b>Example:</b>               | <code>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/filter?summary=false</code>   |
| <b>Example XML response:</b>  | <pre> &lt;Filters&gt;   &lt;Filter&gt;     &lt;changeStamp&gt;0&lt;/changeStamp&gt;     &lt;description&gt;this is an AUTHOR filter!&lt;/description&gt;     &lt;keywords&gt;       &lt;keyword&gt;author name&lt;/keyword&gt;       &lt;keyword&gt;author userid&lt;/keyword&gt;     &lt;/keywords&gt;     &lt;name&gt;filter1&lt;/name&gt;     &lt;refURL&gt;       http://[ServerIP]:[Port]/ccp-webapp/ccp/filter/[id]     &lt;/refURL&gt;     &lt;rule&gt;1&lt;/rule&gt;     &lt;type&gt;3&lt;/type&gt;   &lt;/Filter&gt;   &lt;Filter&gt;     &lt;changeStamp&gt;0&lt;/changeStamp&gt;     &lt;description&gt;this is a SCRIPT filter!&lt;/description&gt;     &lt;name&gt;filter2&lt;/name&gt;     &lt;ScriptFileName&gt;me.groovy&lt;/ScriptFileName&gt;     &lt;scriptContent&gt;       &lt;scriptContentRefURL&gt;         http://[ServerIP]:[Port]/ccp-webapp/ccp/filter/[id]/scriptcontent       &lt;/scriptContentRefURL&gt;     &lt;/scriptContent&gt;     &lt;refURL&gt;       http://[ServerIP]:[Port]/ccp-webapp/ccp/filter/[id]     &lt;/refURL&gt;     &lt;type&gt;4&lt;/type&gt;   &lt;/Filter&gt; &lt;/Filters&gt; </pre> |
| <b>HTTP response headers:</b> | <pre> http/1.1 200 OK Content-Type: application/xml Transfer-Encoding: chunked Date: Tue, 12 Jan 2010 16:47:58 GMT </pre>   |

## GET

Returns the data for a single filter.

|                     |  |
|---------------------|--|
| <b>URL:</b>         | <code>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/filter/&lt;ID Variables&gt;</code> |
| <b>HTTP method:</b> | GET  |

|   |   |
|---|---|
| <b>Example:</b>                                   | <code>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/filter/100036</code>  |
| <b>Parameters:</b>                                | See <a href="#">Filter API Parameters</a> , on page 87.   |
| <b>Example XML response for an author filter:</b> | <pre> &lt;Filter&gt;   &lt;changeStamp&gt;0&lt;/changeStamp&gt;   &lt;name&gt;filter1&lt;/name&gt;   &lt;refURL&gt;     http://[ServerIP]:[Port]/ccp-webapp/ccp/filter/[id]   &lt;/refURL&gt;   &lt;type&gt;3&lt;/type&gt;   &lt;keywords&gt;     &lt;keyword&gt;author name&lt;/keyword&gt;     &lt;keyword&gt;author userid&lt;/keyword&gt;   &lt;/keywords&gt;   &lt;rule&gt;1&lt;/rule&gt; &lt;/Filter&gt; </pre>   |
| <b>Example XML response for a script filter:</b>  | <pre> &lt;Filter&gt;   &lt;changeStamp&gt;0&lt;/changeStamp&gt;   &lt;description&gt;     this is a SCRIPT filter!   &lt;/description&gt;   &lt;name&gt;filter2&lt;/name&gt;   &lt;scriptFileName&gt;me.groovy&lt;/scriptFileName&gt;   &lt;scriptContentRefURL&gt;     http://[ServerIP]:[Port]/ccp-webapp/ccp/filter/[id]/scriptcontent   &lt;/scriptContentRefURL&gt;   &lt;refURL&gt;     http://[ServerIP]:[Port]/ccp-webapp/ccp/filter/[id]   &lt;/refURL&gt;   &lt;type&gt;4&lt;/type&gt; &lt;/Filter&gt; </pre> |
| <b>HTTP response headers:</b>                     | <pre> http/1.1 200 OK Content-Type: application/xml Transfer-Encoding: chunked Date: Tue, 12 Jan 2010 16:50:46 GMT </pre>   |

## GET (Script Content)

This request is valid for script filters only and returns the script content.

|                                      |  |
|--------------------------------------|--|
| <b>URL:</b>                          | <code>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/filter/(id)/scriptcontent</code>             |
| <b>HTTP method:</b>                  | GET  |
| <b>Response for a script filter:</b> | <pre> function{   if((a&lt;b) &amp;&amp; (c&gt;d) ) {     after=before;     before=after;   } } </pre> |

## PUT

Updates an existing filter.

|   |  |
|---|--|
| <b>URL:</b>   | http://<ServerIP>:<Port>/ccp-webapp/ccp/filter/<ID Variables>  |
| <b>HTTP method:</b>                                     | PUT  |
| <b>Parameters:</b>                                      | See <a href="#">Filter API Parameters</a> , on page 87.  |
| <b>Example XML request payload for Bayesian filter:</b> | <pre>&lt;Filter&gt;   &lt;name&gt;Bayesian2&lt;/name&gt;   &lt;description&gt;a different description&lt;/description&gt;   &lt;changeStamp&gt;0&lt;/changeStamp&gt; &lt;type&gt;2&lt;/type&gt; &lt;/Filter&gt;</pre>  |
| <b>Example XML request payload for author filter:</b>   | <pre>&lt;Filter&gt;   &lt;name&gt;String&lt;/name&gt;   &lt;description&gt;String&lt;/description&gt;   &lt;type&gt;3&lt;/type&gt;   &lt;changeStamp&gt;12345&lt;/changeStamp&gt;   &lt;keywords&gt;     &lt;keyword&gt;author name&lt;/keyword&gt;     &lt;keyword&gt;author userid&lt;/keyword&gt;     &lt;keyword&gt;alternate form on author name&lt;/keyword&gt;   &lt;/keywords&gt;   &lt;rule&gt;1&lt;/rule&gt; &lt;/Filter&gt;</pre> |
| <b>Example XML request payload for script filter:</b>   | <pre>&lt;Filter&gt;   &lt;changeStamp&gt;12345&lt;/changeStamp&gt;   &lt;name&gt;String&lt;/name&gt;   &lt;description&gt;A new description&lt;/description&gt;   &lt;type&gt;4&lt;/type&gt;   &lt;scriptFileName&gt;me.groovy&lt;/scriptFileName&gt;   &lt;scriptContent&gt;     &lt;![CDATA[ put the updated text of the script       here.]]&gt;   &lt;/scriptContent&gt; &lt;/Filter&gt;</pre>   |
| <b>HTTP response headers:</b>                           | <pre>http/1.1 200 OK Content-Type: text/plain Content-Length: 0 Date: Thu, 14 Jan 2010 15:49:17 GMT</pre>  |

## Filter API Parameters

Parameters are optional unless otherwise noted.

| Parameter        | Description   | Notes  |
|------------------|---|--|
| name             | The name of the filter. Must be unique.                           | Required for POST.   |
| description      | A description of the filter.                                      |  |
| keywords/keyword | List of keywords for include or exclude rule filtering.           | Currently used only for author filters.  |
| changeStamp      | The change stamp of the filter record.<br>Integer. Defaults to 0. | A <a href="#">changeStamp</a> is required for PUT (update API).<br>changeStamp is returned in GET. |

| Parameter           | Description   | Notes   |
|---------------------|---|---|
| refURL              | The URL requested.  |   |
| rule                | Integer. Sets the filter rule. Values can be: <ul style="list-style-type: none"> <li>• 0 = UNKNOWN</li> <li>• 1 = EXCLUDE: exclude this author from the campaign results.</li> <li>• 2 = INCLUDE: include this author from the campaign results.</li> </ul> | Required only for author filters.   |
| scriptContent       | Contains the text of script filter. Only one script filter is allowed per filter. Currently only supports groovy scripts.   | Used only for script filters.<br>Required for POST (create) for script filters.   |
| scriptContentRefURL | Pointer to the script content.  | Used only for script filters and is returned by GET for a script filter id.<br><br>To review the script content, use <a href="#">GET (Script Content)</a> , on page 86. |
| scriptFileName      | Should be the same as the uploaded script filter file name.   | Used only for script filters.<br>Required for POST (create) for script filters.   |
| type                | The type of filter. Must be one of: <ul style="list-style-type: none"> <li>• 0 = UNKNOWN</li> <li>• 1 = WORD_COUNT_LESS_THAN_SIX</li> <li>• 2 = Bayesian</li> <li>• 3 = Author</li> <li>• 4 = SCRIPT</li> </ul>   | Required for POST (create).   |

## About Script Filters

Script filters are a special type of SocialMiner filter that can execute arbitrary code and modify a social contact. The campaign subsystem sends a filter request to the filter subsystem for a filter of this type. The configured script indicates what script is to be run to filter the social contact.

### Related Topics

- [Script Binding, on page 89](#)
- [Develop and Test Script Filters, on page 90](#)
- [Script Filter Security, on page 92](#)
- [Sample Script Filters, on page 93](#)

## Script Binding

When it is run, a script filter has access to all of the objects in its binding. A binding is a map of variable names to objects that are passed to the script engine and that can be accessed and modified (by name) from the script. This defines the API available to the script.

The objects in the binding are:

- **log**—an object that can be called like a method and passed a string that will be logged by the filter. The output is logged in the application logs (with the name of the script name identifying the message) and is returned in the XML response to the filter results API call.
- **restClient**—an instance of `groovyx.net.http.RESTClient` that can be used by the script to make REST calls to third party APIs. This object is specific to Groovy; it is more of a convenience for making REST calls.
- **socialContact**—An object of type `ScriptFilterSocialContact` that has fields exposed. A copy of this object will be placed in the binding when the script is called. When the script exits, if the information in the copy has been modified, the `SocialContact` modifications are retained to the data store. The exposed fields are:
  - **author**
  - **categories**
  - **description**
  - **\* publishedDate**
  - **tags**—Tags are of type `List<String>`. Duplicate tags are removed during the post-processing stages of the script engine.
  - **title**
  - **\* link**
  - **\* sourceLink**

Note that fields marked with **\*** are read-only. Changes to these fields are not retained.

Filter execution is multi-threaded, the order is non-deterministic, and the campaign status is set after filtering is complete. For these reasons, the `filterResults` and `campaignResults` fields are omitted.

## Develop and Test Script Filters

You can test scripts as you development them by using the filter results API and passing it a social contact id. Doing this will run the filter on the social contact and return the results in XML, along with the output of any logs or exceptions output by the script.



### Note

Errors can result if an editor such as Notepad is used to edit a script written in a language containing multi-byte characters. Eclipse and Notepad++ are the most reliable editors to use for editing scripts.

The procedure to test script filters during development is as follows:

### Procedure

- Step 1** Run a Campaign results [GET, on page 31](#) to obtain a social contact link. Each <entry> has a <link rel="socialcontact" href="socialcontactRefURL">.
- Step 2** Run a Filters [GET \(List\), on page 84](#) to obtain the refURL of the script filter you want to test.
- Step 3** Run a Filter Results [GET, on page 97](#) and add the refURL of the social contact as a parameter.

### Example:

```
http://<ServerIP>:<Port>/ccp-webapp/ccp/filter/
103105/results?socialContact=
http://<ServerIP>:<Port>/ccp-webapp/ccp/socialcontact/
AB1C35141000013200000F450A568DD2.
```

- Step 4** Enter that URL in the address bar of a browser to view the XML filter result.

### Example:

The <logBuffer> is all the output from the script, including logging and exceptions.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <FilterResult>
  <logBuffer>
Bad script.
Security rules violation exception: startup failed:
General error during canonicalization:
Indirect import checks prevents
usage of expression java.lang.SecurityException:
Indirect import checks prevents usage of expression
at org.codehaus.groovy.control.customizers.
SecureASTCustomizer$SecuringCode
Visitor.assertExpressionAuthorized
(SecureASTCustomizer.java:682)
at org.codehaus.groovy.control.customizers.
SecureASTCustomizer
$SecuringCodeVisitor.visitConstructorCallExpression
SecureASTCustomizer.java:845)
at org.codehaus.groovy.ast.expr.
ConstructorCallExpression.visit
(ConstructorCallExpression.java:43)
at org.codehaus.groovy.control.customizers.
SecureASTCustomizer
$SecuringCodeVisitor.visitThrowStatement
(SecureASTCustomizer.java:804)
at org.codehaus.groovy.ast.stmt.ThrowStatement.visit
```

```

(ThrowStatement.java:41)
at org.codehaus.groovy.control.customizers.
SecureASTCustomizer
$SecuringCodeVisitor.visitBlockStatement
(SecureASTCustomizer.java:705)
at org.codehaus.groovy.ast.stmt.BlockStatement.
visit(BlockStatement.java:69)
at org.codehaus.groovy.control.customizers.
SecureASTCustomizer.call
(SecureASTCustomizer.java:549)
at org.codehaus.groovy.control.CompilationUnit.
applyToPrimaryClassNodes
(CompilationUnit.java:957)
at org.codehaus.groovy.control.CompilationUnit.
doPhaseOperation
(CompilationUnit.java:542)
at org.codehaus.groovy.control.CompilationUnit.
processPhaseOperations
(CompilationUnit.java:520)
at org.codehaus.groovy.control.CompilationUnit.
compile(CompilationUnit.java:497)
at groovy.lang.GroovyClassLoader.doParseClass
(GroovyClassLoader.java:306)
at groovy.lang.GroovyClassLoader.parseClass
(GroovyClassLoader.java:287)
at groovy.util.GroovyScriptEngine$ScriptClassLoader.
parseClass
(GroovyScriptEngine.java:197)
at groovy.lang.GroovyClassLoader.parseClass
(GroovyClassLoader.java:267)
at groovy.lang.GroovyClassLoader.parseClass
(GroovyClassLoader.java:214)
at groovy.util.GroovyScriptEngine.loadScriptByName
(GroovyScriptEngine.java:470)
at groovy.util.GroovyScriptEngine.createScript
(GroovyScriptEngine.java:539)
at groovy.util.GroovyScriptEngine.run
(GroovyScriptEngine.java:526)
at com.cisco.ccbu.ccp.filter.ScriptFilter.
executeFilterOnSocialContact
(ScriptFilter.java:148)
at com.cisco.ccbu.ccp.filter.FilterManager.
executeGenericFilter
(FilterManager.java:688)
at com.cisco.ccbu.ccp.filter.FilterManager.
applyFilterOnSocialContact
(FilterManager.java:497)
at com.cisco.ccbu.ccp.filter.FilterManager.
applyFilterOnSocialContact
(FilterManager.java:413)
at com.cisco.ccbu.ccp.filter.FilterSubsystem.
executeFilterOnSocialContact
(FilterSubsystem.java:356)
at com.cisco.ccbu.ccp.filter.FilterSubsystem.
handleFilterSocialContactRequest
(FilterSubsystem.java:334)
at com.cisco.ccbu.ccp.filter.FilterSubsystem.
handleMessage
(FilterSubsystem.java:130)
at com.cisco.ccbu.ccp.filter.messaging.FilterMsgHandler.
handleMessage
(FilterMsgHandler.java:22)
at com.cisco.ccbu.infra.msg.BaseMessage$Handler.
handleMessageInternal
(BaseMessage.java:1197)
at com.cisco.ccbu.infra.msg.BaseMessage$Handler.
handleMessage
(BaseMessage.java:1175)
at com.cisco.ccbu.infra.msg.MSGHolder.handleImpl
(MSGHolder.java:322)
at com.cisco.ccbu.infra.msg.MSGDispatcher$Hook.handle
(MSGDispatcher.java:2976)

```

```

at com.cisco.ccbu.infra.msg.MSGDispatcher$DispatchRunnable.
handleMessage
(MSGDispatcher.java:3232)
at com.cisco.ccbu.infra.msg.MSGDispatcher$DispatchRunnable.run
(MSGDispatcher.java:3262)
at com.cisco.ccbu.infra.threads.InstrumentedRunnable.run
(InstrumentedRunnable.java:88)
at java.util.concurrent.ThreadPoolExecutor$Worker.runTask
(ThreadPoolExecutor.java:886)
at java.util.concurrent.ThreadPoolExecutor$Worker.run
(ThreadPoolExecutor.java:908)
at java.lang.Thread.run(Thread.java:619)
at com.cisco.ccbu.infra.threads.ThreadPoolThread.run
(ThreadPoolThread.java:164)
Caused by: java.lang.SecurityException: Importing
[java.lang.NullPointerException] is not allowed
at org.codehaus.groovy.control.customizers.SecureASTCustomizer.
assertImportIsAllowed
(SecureASTCustomizer.java:574)
at org.codehaus.groovy.control.customizers.SecureASTCustomizer.
access$800
(SecureASTCustomizer.java:121)
at org.codehaus.groovy.control.customizers.SecureASTCustomizer
$SecuringCodeVisitor.assertExpressionAuthorized
(SecureASTCustomizer.java:664) ... 38 more 1 error
</logBuffer>

<refURL>
http://[ServerIP]:[Port]/ccp-webapp/ccp/
filter/103105/results</refURL>
<result>100</result>
<socialContact>
http://[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/
AB1C35141000013200000F450A568DD2
</socialContact>
</FilterResult>

```

---

## Script Filter Security

SocialMiner imposes restrictions on Groovy script code to ensure the security and integrity of the system and data.

For example, scripts are restricted from:

- Shutting down the system (system.exit()).
- Calling native Java methods (so as not to corrupt memory).
- Executing for longer than 30 seconds.
- Accessing the SocialMiner file system.
- Executing certain SQL commands.

A script that violates these restrictions will upload but will have no impact on the social contacts in the campaign to which it is applied.

In addition to [Script Binding](#), you can create objects from these classes:

- java.lang.Object
- java.lang.Boolean



- java.lang.Integer
- java.lang.Float
- java.lang.Short
- java.lang.Long
- java.lang.Double
- java.util.Date
- java.util.List
- java.util.Map
- java.util.Set
- java.util.Collections
- java.lang.String
- java.lang.StringBuilder
- java.util.TreeSet
- java.util.Vector
- java.util.LinkedHashSet
- java.util.LinkedList
- java.util.Stack
- java.util.ArraySet
- java.util.Arrays
- java.util.HashMap
- java.util.SortedMap
- java.util.TreeMap
- java.util.LinkedHashMap
- org.apache.commons.lang.StringUtils
- org.apache.commons.lang.Validate
- groovyx.net.http.HttpResponseDecorator
- java.util.Random
- java.math.\*

## Sample Script Filters

You can create script filters to change or add to the content of social contacts and to call external web services. For example, you can create a script to translate text to another language, to analyze sentiment ([opendover](#)), or to recognize trends ([Google Prediction](#)).

As of release 8.5(5), SocialMiner script filters use the [GroovyScriptEngine](#). SocialMiner runs scripts with Groovy 1.8.

## Script Filter for Social Contact Modification

```
/*
 * Example script that modifies a social contact
 *
 * This script will demonstrate the modification of a social contact.
 * For the full list of script filter fields, see ->
 * http://cvp/display/ccpdev/Filter+Script+API
 */
//Set the author
socialContact.author = "John Doe"

//Set the title
socialContact.title = "New Title"

//Set the description
socialContact.description = "This is a socialContact"

//Set the categories. Takes a list of strings.
socialContact.categories = ["category_1", "category_2", "category_3"]

//Set the tags. Alternatively, you can use the Java syntax as well.
//NOTE: duplicate tags will be removed when the socialContact is saved.
//NOTE: setting tags like so will replace any existing tags.
// to append tags see below.
socialContact.tags = ["tag1", "tag1", "tag2"]

//Append new_tag to tags
def tags = socialContact.tags;
tags += "tag3"
socialContact.tags = tags;

//A shorter way
socialContact.tags.add("tag4")

//Alternatively,
socialContact.tags += "tag5"

//Or even
socialContact.tags += ["tag6", "tag7"]

//Log the author, title, description, categories, and tags using getter methods.
log "Author is " + socialContact.author //should be "John Doe"
log "Title is " + socialContact.title //should be "New Title"
log "Description is " + socialContact.description //should be "This is a socialContact"
log "Categories are" + socialContact.categories //should
be["category_1","category_2","category_3"]
log "Tags are" + socialContact.tags //should be [ "tag1", "tag1", "tag2", "tag3", "tag4",
"tag5",
"tag6", "tag7"]

//however, keep in mind that duplicates will be removed when saved

//Finally, return
```

## Script Filter for Klout Score Generation

This is an example of a script filter that generates a Klout score for social contacts that are gathered from Twitter Account feeds:

```
def KLOUT_KEY = <klout_key>;

restClient.setProxy("161.44.248.59", 80, null);
```

```
String user = socialContact.getAuthor();
int space = user.indexOf(' ');
if (space != -1)
    user = user.substring(0, space);

log "user = " + user;

def resp = restClient.get( uri: "http://api.klout.com/1/klout.xml", query :
[key: KLOUT_KEY, users: user]);

if (!resp.isSuccess()) {
    log ("Request Failed");
}
else {
    log "Influence: " + resp.data.user.kscore;
    boolean influential = (Double.parseDouble(resp.data.user.kscore as String) > 20.0);
    log "User is influential" + influential;
    if (influential){
        def tags = socialContact.tags;
        tags += "influential" as String;
        log "New tags: " + tags;
        socialContact.tags = tags;
    }
    return influential?100:0;
}
```





## Filter Results

The Filter Results API allows you to get the results of a specified filter for analysis of the text passed to it.

- [Filter Results API Commands](#), page 97

## Filter Results API Commands

This section describes the supported command (GET) for the filter results API and the parameters for that command.

### GET

Get results for the specified filter.

|                              |  |
|------------------------------|--|
| <b>URL:</b>                  | http://<ServerIP>:<Port>/ccp-webapp/ccp/filter/<ID Variables>/results  |
| <b>HTTP method:</b>          | GET  |
| <b>URL parameters:</b>       | <ul style="list-style-type: none"><li>• <b>document:</b> the text that the configured filter analyzes.</li><li>• <b>socialContact:</b> the refURL of the social contact that the configured filter analyzes.</li></ul> <p>It is valid to provide either document or social contact, but invalid to provide both or neither.</p>  |
| <b>Example XML response:</b> | <p>The filter results are returned as a single &lt;FilterResult&gt; element that contains five required child elements.</p> <p><b>FilterResult:</b> the container for the result.</p> <ul style="list-style-type: none"><li>• <b>document:</b> the text passed to the filter for analysis.</li><li>• <b>logBuffer:</b> the aggregation of the social contact filter log output.</li><li>• <b>refURL:</b> the URL of the filter results request.</li><li>• <b>result:</b> the result of the filter analysis expressed as an integer from 1–100.</li></ul> |

- **socialContact**: the refURL of the social contact that was passed to the filter for analysis.

```
<FilterResult>
  <refURL>
    http://[ServerIP]:[Port]/ccp-webapp/ccp/
    filter/[id]/results
  </refURL>
  <result>100</result>
  <document>
    The text that was passed to the filter for analysis.
  </document>
  <socialContact>
    The refURL of the social contact that was passed to the
    filter for analysis.
  </socialContact>
  <logBuffer>
    The social contact filter log output.
  </logBuffer>
</FilterResult>
```



## Notification Rule

The Notification rule API allows you to configure notifications that are sent when a specific tag is added to a contact in a specific campaign.



### Note

Only the administrator created during install can use this API.

This API is represented on the SocialMiner user interface in the Notifications panel.

There are four types of notification rules: email, IM, http and (connection to) CCE. The parameters to use when creating or updating a notification rule depend on the type. The following table lists the parameters that are applicable to each type.

| Rule type | Required parameters   | Optional parameters  | Notes   |
|-----------|---|--|---|
| email     | name, campaignUrl, tags, type, targets                              | description, subject, body                                     |   |
| im        | name, campaignUrl, tags, type, targets                              | description, body  |   |
| http      | name, campaignUrl, tags, type, httpUrl                              | description, httpUsername, httpPassword, sslVerifyCertificates |   |
| cce       | name, campaignUrl, tags, type, scriptSelector, mediaRoutingDomainId | description  | <p>The scriptSelector is the Dialed Number String or Script Selector from the CCE configuration.</p> <p>The mediaRoutingDomainId is the ID of the selected Media Routing Domain from the CCE configuration. Currently, the only valid value for mediaRoutingDomainId is "1" (for voice callback).</p> |

**Note**

Fields not relevant to a given notification type will be ignored. For example, a body specified in a http notification rule will be ignored.

**Note**

You must configure an Email (SMTP) Server before notification can be sent through email. You must configure an XMPP server before IM notifications can be sent. Connection to CCE Notifications send a request to CCE with media routing information.

- [Notification API Commands, page 100](#)
- [Notification Keywords in Email and IM , page 106](#)
- [HTTP Notifications, page 107](#)

## Notification API Commands

This section describes the supported commands for the Notification API and the parameters for those commands.

### POST

Creates a notification rule.

|   |  |
|---|--|
| <b>URL:</b>                                 | http://<ServerIP>:<Port>/ccp-webapp/ccp/notificationrule   |
| <b>HTTP method:</b>                         | POST   |
| <b>Example XML request payload (email):</b> | <pre>&lt;NotificationRule&gt;   &lt;name&gt;test&lt;/name&gt;   &lt;description&gt;     this is the description   &lt;/description&gt;   &lt;campaignUrl&gt;     http://[ServerIP]:[Port]/ccp-webapp/ccp/     campaign/MyTestCampaign   &lt;/campaignUrl&gt;   &lt;tags&gt;     &lt;tag&gt;test&lt;/tag&gt;     &lt;tag&gt;cisco&lt;/tag&gt;   &lt;/tags&gt;   &lt;targets&gt;     &lt;target&gt;test@cisco.com&lt;/target&gt;     &lt;target&gt;cisco@cisco.com&lt;/target&gt;   &lt;/targets&gt;   &lt;type&gt;email&lt;/type&gt;   &lt;subject&gt;Notification Rules&lt;/subject&gt;   &lt;body&gt;Click on this link.&lt;/body&gt; &lt;/NotificationRule&gt;</pre> |
| <b>Example XML request</b>                  | <pre>&lt;NotificationRule&gt;   &lt;name&gt;test&lt;/name&gt;</pre>  |



|   |  |
|---|--|
| <b>payload (http):</b>                    | <pre> &lt;description&gt;   this is the description &lt;/description&gt; &lt;campaignUrl&gt;   http://[ServerIP]:[Port]/ccp-webapp/ccp/   campaign/MyTestCampaign &lt;/campaignUrl&gt; &lt;tags&gt;   &lt;tag&gt;test&lt;/tag&gt;   &lt;tag&gt;cisco&lt;/tag&gt; &lt;/tags&gt; &lt;type&gt;http&lt;/type&gt; &lt;httpUrl&gt;   http://someserver/notification/handler &lt;/httpUrl&gt; &lt;httpUsername&gt;username&lt;/httpUsername&gt; &lt;httpPassword&gt;password&lt;/httpPassword&gt; &lt;sslVerifyCertificates&gt;   true &lt;/sslVerifyCertificates&gt; &lt;/NotificationRule&gt; </pre>  |
| <b>Example XML request payload (CCE):</b> | <pre> &lt;NotificationRule&gt;   &lt;name&gt;test&lt;/name&gt;   &lt;description&gt;this is the description&lt;/description&gt;   &lt;campaignUrl&gt;http://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/MyTestCampaign &lt;/campaignUrl&gt;   &lt;tags&gt;     &lt;tag&gt;test&lt;/tag&gt;     &lt;tag&gt;cisco&lt;/tag&gt;   &lt;/tags&gt;   &lt;type&gt;cce&lt;/type&gt;   &lt;scriptSelector&gt;allSalesAndService&lt;/scriptSelector&gt;   &lt;mediaRoutingDomains&gt;     &lt;mediaRoutingDomain&gt;       &lt;mediaRoutingDomainId&gt;1&lt;/mediaRoutingDomainId&gt;     &lt;/mediaRoutingDomain&gt;   &lt;/mediaRoutingDomains&gt; &lt;/NotificationRule&gt; </pre> |
| <b>Parameters:</b>                        | See <a href="#">Notification API Parameters</a> , on page 104.   |
| <b>HTTP response headers:</b>             | A <i>201 Created</i> http header is returned on success, along with the REST URL to the new notification rule.   |

## PUT

Updates an existing notification rule.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | <a href="http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/notificationrule/&lt;ID Variables&gt;">http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/notificationrule/&lt;ID Variables&gt;</a> |
| <b>HTTP method:</b> | PUT   |
| <b>Parameters:</b>  | See <a href="#">Notification API Parameters</a> , on page 104. All parameters are optional for the Notification API update operation.   |

## DELETE

|                               |  |
|-------------------------------|--|
| <b>Example XML response:</b>  | <pre> &lt;NotificationRule&gt;   &lt;body&gt;New Contact:&lt;/body&gt;   &lt;campaignUrl&gt;     http://[ServerIP]:[Port]/ccp-webapp/ccp/     campaign/Pushed_Contacts   &lt;/campaignUrl&gt;   &lt;changeStamp&gt;0&lt;/changeStamp&gt;   &lt;name&gt;Push&lt;/name&gt;   &lt;subject&gt;     Notification: New Push Tag applied to Pushed     Contacts Campaign   &lt;/subject&gt;   &lt;tags&gt;     &lt;tag&gt;push&lt;/tag&gt;   &lt;/tags&gt;   &lt;targets&gt;     &lt;target&gt;user@example.com&lt;/target&gt;   &lt;/targets&gt;   &lt;type&gt;email&lt;/type&gt; </pre> |
| <b>HTTP response headers:</b> | A <i>200 OK</i> http header is returned on success.  |

## DELETE

Deletes a notification rule.

|                               |   |
|-------------------------------|---|
| <b>URL:</b>                   | <a href="http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/notificationrule/&lt;ID Variables&gt;">http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/notificationrule/&lt;ID Variables&gt;</a> |
| <b>HTTP method:</b>           | DELETE  |
| <b>HTTP response headers:</b> | A <i>200 OK</i> http header is returned on success.   |

## GET (List)

Lists all notification rules.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | <a href="http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/notificationrule">http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/notificationrule</a> |
| <b>HTTP method:</b> | GET   |

|                               |  |
|-------------------------------|--|
| <b>Example XML response:</b>  | <pre> &lt;NotificationRules&gt;   &lt;NotificationRule&gt;     &lt;body&gt;New Contact:&lt;/body&gt;     &lt;campaignUrl&gt;       http://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/       Pushed_Contacts     &lt;/campaignUrl&gt;     &lt;changeStamp&gt;0&lt;/changeStamp&gt;     &lt;name&gt;Push&lt;/name&gt;     &lt;refURL&gt;       http://[ServerIP]:[Port]/ccp-webapp/ccp/       notificationrule/100010     &lt;/refURL&gt;     &lt;subject&gt;       Notification: New Push Tag applied       to Pushed Contacts Campaign     &lt;/subject&gt;     &lt;tags&gt;       &lt;tag&gt;push&lt;/tag&gt;     &lt;/tags&gt;     &lt;targets&gt;       &lt;target&gt;user@example.com&lt;/target&gt;     &lt;/targets&gt;     &lt;type&gt;email&lt;/type&gt;   &lt;/NotificationRule&gt;   .... &lt;/NotificationRules&gt; </pre> |
| <b>HTTP response headers:</b> | A <i>200 OK</i> http header is returned on success.  |

## GET

Retrieves a specific notification rule.

|                               |  |
|-------------------------------|--|
| <b>URL:</b>                   | http://<ServerIP>:<Port>/ccp-webapp/ccp/notificationrule/<ID Variables>  |
| <b>HTTP method:</b>           | GET  |
| <b>Example XML response:</b>  | <pre> &lt;NotificationRule&gt;   &lt;body&gt;New Contact:&lt;/body&gt;   &lt;campaignUrl&gt;http://[ServerIP]:[Port]/   ccp-webapp/ccp/campaign/Pushed_Contacts&lt;/campaignUrl&gt;   &lt;changeStamp&gt;0&lt;/changeStamp&gt;   &lt;name&gt;Push&lt;/name&gt;   &lt;refURL&gt;http://[ServerIP]:[Port]/   ccp-webapp/ccp/notificationrule/100010&lt;/refURL&gt;   &lt;subject&gt;Notification: New Push Tag applied to Pushed   Contacts Campaign&lt;/subject&gt;   &lt;tags&gt;     &lt;tag&gt;push&lt;/tag&gt;   &lt;/tags&gt;   &lt;targets&gt;     &lt;target&gt;user@example.com&lt;/target&gt;   &lt;/targets&gt;   &lt;type&gt;email&lt;/type&gt; &lt;/NotificationRule&gt; </pre> |
| <b>HTTP response headers:</b> | A <i>200 OK</i> http header is returned on success.  |

## Notification API Parameters

The parameters to use when creating or updating a notification rule depend on the notification type. The following table lists the parameters that are applicable to each type.

Parameters are optional unless otherwise noted.

| Parameter          | Description  | Notes  |
|--------------------|--|--|
| <b>changeStamp</b> | The change stamp of the notification record.                                     | Integer. Defaults to 0.<br>changeStamp is returned in GET.<br>changeStamp is required for PUT (Update API).  |
| <b>name</b>        | The name of the notification rule.   | Required for POST.   |
| <b>campaignUrl</b> | The URL of the campaign.   | String.<br>Required for POST.  |
| <b>description</b> | Description.   | String.<br>Follows SocialMiner standard description naming conventions.  |
| <b>tags/tag</b>    | Tag or list of tags from which the rule is activated and a notification is sent. | String.<br>Required for POST.<br>Maximum of 5.   |
| <b>type</b>        | The type of notification to send out for this rule.                              | Valid values are: <ul style="list-style-type: none"> <li>• <i>email</i> (Notification is sent over email.)</li> <li>• <i>IM</i> (Notification is sent over IM.)</li> <li>• <i>http</i> (Notification is used for chat and is reserved for other developer applications that deliver social contact data to an external application.)</li> <li>• <i>(connection to) CCE</i> (Notification sends a request to CCE with media routing information.)</li> </ul> Required for POST. |

| Parameter                    | Description  | Notes   |
|------------------------------|--|---|
| <b>targets/target</b>        | One or more targets to which the notification rule is sent.  | String.<br>Required for POST.<br>Maximum of 10.   |
| <b>httpUrl</b>               | The URL of a REST API.   | Required for POST.<br>SocialMiner will post details of a social contact to this URL.  |
| <b>subject</b>               | The subject of a notification rule message.  | String.<br>Maximum of 255 characters.   |
| <b>body</b>                  | The body of the notification rule message. The link to the social contact is automatically inserted after the body. Link URL will appear beneath the body. | String.<br>Maximum of 2048 characters.<br>The body may contain reserved keywords in a special syntax.<br>These <a href="#">Notification Keywords in Email and IM</a> will be replaced with values from the social contact. The reserved word syntax takes the form <code>\${KEYWORD}</code> . |
| <b>httpUsername</b>          | Username   | Required if authentication is necessary to use the REST API specified by <b>httpUrl</b> .   |
| <b>httpPassword</b>          | Password   | Required if authentication is necessary to use the REST API specified by <b>httpUrl</b> .   |
| <b>sslVerifyCertificates</b> | Defines if SSL certificate verification will be enabled or disabled for connections made for notification.   | Boolean. Default is True.<br>HTTP only.   |
| <b>scriptSelector</b>        | The Dialed Number String/Script Selector from the CCE configuration.   |   |
| <b>mediaRoutingDomains</b>   | A list of media routing domains, each of which contains the media routing domain Id of the media routing domain from the CCE Configuration.                | Integer, not null.<br>At present, only one Media Routing Domain is supported.   |

# Notification Keywords in Email and IM

The body of an email or IM may contain reserved words in a special syntax of the form `${KEYWORD}`. These key words will be replaced with values from the social contact. Keywords are listed here in uppercase, but they are case-insensitive.

The currently defined keywords are:

- `SC_AUTHOR`—this keyword is replaced with the social contact author.
- `SC_CREATED_DATE`—this keyword is replaced with the social contact created date.
- `SC_DESCRIPTION`—this keyword is replaced with the social contact description.
- `SC_PUBLISHED_DATE`—this keyword is replaced with the social contact published date.
- `SC_SCREEN_URL`—this keyword is replaced with the URL (a live link).
- `SC_SOURCE_TYPE`—this keyword is replaced with the social contact source type, for example: RSS, Twitter stream, Twitter account, or Facebook.
- `SC_TAGS`—this keyword is replaced with the social contact tags.
- `SC_TITLE`—this keyword is replaced with the social contact title.
- `SC_TAGS`—this keyword is replaced with the social contact tags.
- `SC_EXTENSION_FIELD.<fieldname>`—An extension field is additional data for the social contact. You can add a maximum of 100 extension fields, up to one megabyte of information. Like the keyword, the fieldname extension must also be enter in all upper case letters.
- `SC_EXTENSION_FIELDS`—Returns all extension fields for the social contact. They appear in the body in alphabetical order by name in the format *Name: Value*. If no value was defined, you see *Name: (—)*. If there are no extension fields, the variable is removed from the message body.

For example, if the Body is set to:

- The author of this Social Contact is: `${SC_AUTHOR}`.
- The message contents are as follows: `${SC_DESCRIPTION}`.

Then the notification message body would contain:

The author of this Social Contact is: *someAuthorNameIfItWasProvided*.  
The message contents are as follows: *theContentsOfTheSocialContactMessage*.

If the body is set empty (null or blank), then the notification message will contain the `SC_SCREEN_URL` value by default.

Email messages are sent in HTML format and the body contents in the notification rule may contain user entered HTML markup.

IM messages are in text format, not HTML.

# HTTP Notifications

HTTP notifications will post a message with the following body to the URL specified in the notification rule `httpUrl` parameter.

|                     |  |
|---------------------|--|
| <b>HTTP method:</b> | POST   |
| <b>Fields:</b>      | <ul style="list-style-type: none"> <li>• <b>author:</b> author of the social contact.</li> <li>• <b>description:</b> body of the social contact.</li> <li>• <b>id:</b> datastore id of the social contact.</li> <li>• <b>link:</b> unique id of the original social contact (RSS ID or Twitter Id or Facebook Id, etc).</li> <li>• <b>notificationTag:</b> the tag that fired off the notification rule.</li> <li>• <b>publishedDate:</b> the publish date of the document.</li> <li>• <b>refURL:</b> the REST id of the social contact. Applications can do a GET on this URL to get social contact detail.</li> <li>• <b>screenPopUrl:</b> URL used to access the social contact in the SocialMiner's UI.</li> <li>• <b>status:</b> string (case-sensitive) One of: <ul style="list-style-type: none"> <li>◦ unread: The default state of a new social contact.</li> <li>◦ reserved: Reserved to be handled.</li> <li>◦ handled: This social contact has been handled and no further action is required.</li> <li>◦ discarded: This social contact does not require a response and is filed in the recycle bin.</li> <li>◦ queued: The callback request was successfully submitted to the contact center for routing.</li> </ul> </li> <li>• <b>statusTimestamp:</b> timestamp of the last status update.</li> <li>• <b>statusUserId:</b> id of the agent who updated the status (initially this is empty).</li> <li>• <b>tags/tag:</b> one or more tags associated with the social contact—normally this is optional but for CCX integration this should contain the routing tag.</li> <li>• <b>extensionFields/extensionField:</b> A collection of custom name and value pairs. The person submitting the social contact may specify up to 100 pairs and the entire collection can contain up to one megabyte of information.</li> <li>• <b>title:</b> title of the social contact.</li> <li>• <b>replyTemplateRefURL:</b> link to the reply template configuration. Present if reply template is configured for the contact.</li> </ul> |

|   |  |
|---|--|
|   | <ul style="list-style-type: none"> <li>• <b>replyTemplateURL:</b> link to the reply template. Present if reply template is configured for the contact.</li> <li>• <b>replyType:</b> expected reply format (currently web, chat, or email). The URI below will be in a format compatible with the reply type.</li> <li>• <b>URI:</b> for chat room.</li> </ul>  |
| <b>Example http notification message:</b> | <pre> &lt;SocialContact&gt; &lt;author&gt;David Dahlquist&lt;/author&gt; &lt;description&gt;New app and video sharing service, Thwapr,   helps overcome the iPhone's video sharing   limitations letting you easily capture and share   videos and photos with many types of mobile   devices.&lt;/description&gt; &lt;id&gt;073D0E871000012E0000ED8B0A568DDF&lt;/id&gt; &lt;link&gt;   http://rss.macworld.com/click.phdo?   i=b942ba3fe59d99b27b08996ad8a9f06a &lt;/link&gt; &lt;notificationTag&gt;ccx:sales&lt;/notificationTag&gt; &lt;publishedDate&gt;1297201140000&lt;/publishedDate&gt; &lt;refURL&gt;   https://[ServerIP]:[Port]/ccp-webapp/   ccp/socialcontact/   073D0E871000012E0000ED8B0A568DDF &lt;/refURL&gt; &lt;screenPopUrl&gt;   http://[ServerIP]:[Port]/results.jsp?   scID=461E5C541000012F000027650A568DF5&amp;   campaignID=httpNotificationCampaign-01465-   00000000000057&lt;/screenPopUrl&gt; &lt;status&gt;unread&lt;/status&gt; &lt;statusTimestamp&gt;1302551491320&lt;/statusTimestamp&gt; &lt;statusUserId&gt;admin&lt;/statusUserId&gt; &lt;tags&gt;   &lt;tag&gt;ccx:sales&lt;/tag&gt;   &lt;tag&gt;ccx:support&lt;/tag&gt; &lt;/tags&gt; &lt;extensionFields&gt;   &lt;extensionField&gt;     &lt;name&gt;accountNumber&lt;/name&gt;     &lt;value&gt;6722392&lt;/value&gt;   &lt;/extensionField&gt;   &lt;extensionField&gt;     &lt;name&gt;remarks&lt;/name&gt;     &lt;value&gt;My CRS-3 is not cooling enough&lt;/value&gt;   &lt;/extensionField&gt; &lt;/extensionFields&gt; &lt;title&gt;Default Entry Title -01465-00000000000032&lt;/title&gt; &lt;/SocialContact&gt; </pre> |
| <b>HTTP response headers:</b>             | A 200 OK http header is returned on success.   |





## Predefined Response

The predefined response API is used to define the sets of predefined responses to the common questions and greeting messages to a chat contact.

- [Create Predefined Response](#), page 109
- [Get Predefined Response](#), page 110
- [List Predefined Response](#), page 111
- [List Predefined Response \(By Group\)](#), page 111
- [Update Predefined Response](#), page 112
- [Delete Predefined Response](#), page 112

### Create Predefined Response

This API creates a predefined response.



#### Note

- A maximum of 500 predefined responses can be created in the system.
- The response text can be up to 2000 characters with UTF8 encoding.
- Group names can be up to 100 characters with a maximum of 10 groups per response.
- Title names can be up to 100 characters.
- At least one group name has to be specified in a response.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/predefinedResponse/ |
| <b>HTTP method:</b> | POST  |

|                                     |   |
|-------------------------------------|---|
| <b>Example XML request payload:</b> | <pre>&lt;response&gt;   &lt;title&gt;A summary of the response&lt;/title&gt;   &lt;text&gt;The text of the response. This can be up to 2000 utf8 characters long&lt;/text&gt;   &lt;groups&gt;     &lt;group&gt;group 1&lt;/group&gt;     &lt;group&gt;group 2&lt;/group&gt;     &lt;group&gt;group 3&lt;/group&gt;   &lt;/groups&gt; &lt;/response&gt;</pre> |
| <b>Parameters:</b>                  | <p><b>title</b> (required) : a short summary of the response.</p> <p><b>text</b> (required) : the text of the response.</p> <p><b>groups</b> (required) : the list of groups associated with the response .</p>   |
| <b>HTTP response headers:</b>       | <p>201 Created (the created URL is returned with the response)</p> <p>400 Bad request (if the input is not as per the defined criteria)</p> <p>For more information, see <a href="#">HTTP Responses</a>.</p>  |

## Get Predefined Response

This API retrieves a predefined response from the system.

|                               |   |
|-------------------------------|---|
| <b>URL:</b>                   | http://<ServerIP>:<Port>/ccp-webapp/ccp/predefinedResponse/<id>   |
| <b>HTTP method:</b>           | GET   |
| <b>Example XML response:</b>  | <pre>&lt;response&gt;   &lt;id&gt;The GUID generated by the API for this response&lt;/id&gt;   &lt;changeStamp&gt;2&lt;/changeStamp&gt;   &lt;title&gt;The title of the response.&lt;/title&gt;   &lt;text&gt;The text of the response.&lt;/text&gt;    &lt;refURL&gt;http://&lt;server&gt;:&lt;serverport&gt;/ccp-webapp/ccp/response/ (id)&lt;/refURL&gt;    &lt;groups&gt;     &lt;group&gt;group 1&lt;/group&gt;     &lt;group&gt;group 2&lt;/group&gt;     &lt;group&gt;group 3&lt;/group&gt;     &lt;group&gt;group n&lt;/group&gt;   &lt;/groups&gt; &lt;/response&gt;</pre> |
| <b>Elements:</b>              | <p><b>title</b> : a short summary of the response.</p> <p><b>text</b> : the text of the response.</p> <p><b>groups</b> : the list of groups associated with the response.</p> <p><b>id</b> : contains the GUID identifying a predefined response.</p>   |
| <b>HTTP response headers:</b> | <p>201 Created</p> <p>404 Not found</p> <p>For more information, see <a href="#">HTTP Responses</a>.</p>  |

## List Predefined Response

This API lists a set of predefined responses from the system.

|                               |  |
|-------------------------------|--|
| <b>URL:</b>                   | http://<ServerIP>:<Port>/ccp-webapp/ccp/predefinedResponse   |
| <b>HTTP method:</b>           | GET  |
| <b>Example XML response:</b>  | <pre> &lt;responses&gt;   &lt;response&gt;     &lt;id&gt;The GUID generated by the API for this response&lt;/id&gt;     &lt;title&gt;Title&lt;/title&gt;     &lt;text&gt;The text of the response.&lt;/text&gt;      &lt;refURL&gt;http://&lt;server&gt;:&lt;serverport&gt;/ccp-webapp/ccp/response/ (id)&lt;/refURL&gt;      &lt;groups&gt;       &lt;group&gt;group 1&lt;/group&gt;       &lt;group&gt;group 2&lt;/group&gt;     &lt;/groups&gt;   &lt;/response&gt;   &lt;response&gt;     &lt;id&gt;The GUID generated by the API for this response&lt;/id&gt;     &lt;title&gt;Title&lt;/title&gt;     &lt;text&gt;Another text of a response. &lt;/text&gt;      &lt;refURL&gt;http://&lt;server&gt;:&lt;serverport&gt;/ccp-webapp/ccp/response/ (id)&lt;/refURL&gt;      &lt;groups&gt;       &lt;group&gt;group 3&lt;/group&gt;       &lt;group&gt;group n&lt;/group&gt;     &lt;/groups&gt;   &lt;/response&gt; &lt;/responses&gt; </pre> |
| <b>HTTP response headers:</b> | 200 OK<br><br>For more information, see <a href="#">HTTP Responses</a> .   |

## List Predefined Response (By Group)

This API retrieves a set of predefined responses by the defined group.



### Note

You can mention a maximum of 10 groups in the parameters. More than 10 groups in the parameter will result in **400 BAD REQUEST**.

|                               |   |
|-------------------------------|---|
| <b>URL:</b>                   | http://<ServerIP>:<Port>/ccp-webapp/ccp/predefinedResponse?group=<group1>&group=<group2>        |
| <b>HTTP method:</b>           | GET   |
| <b>HTTP response headers:</b> | 200 OK<br><br>400 Bad request<br><br>For more information, see <a href="#">HTTP Responses</a> . |

## Update Predefined Response

This API partially or completely updates the predefined response. It can update any specific predefined response as per the given input parameter.



### Note

- Any field updated in the request will completely replace the existing values stored in the database.
- changeStamp* provided by the server should not be altered by the clients, and should be sent in the UPDATE request with other fields to be updated.

|                                     |  |
|-------------------------------------|--|
| <b>URL:</b>                         | http://<ServerIP>:<Port>/ccp-webapp/ccp/predefinedResponse/<id>  |
| <b>HTTP method:</b>                 | PUT  |
| <b>Example XML request payload:</b> | <pre>&lt;response&gt;   &lt;title&gt;A summary of the response&lt;/title&gt;   &lt;changeStamp&gt;3&lt;/changeStamp&gt;   &lt;text&gt;The text of the response.&lt;/text&gt;   &lt;groups&gt;     &lt;group&gt;group 1&lt;/group&gt;     &lt;group&gt;group 2&lt;/group&gt;     &lt;group&gt;group 3&lt;/group&gt;   &lt;/groups&gt; &lt;/response&gt; &lt;response&gt;   &lt;title&gt;A summary of the response&lt;/title&gt;   &lt;changeStamp&gt;3&lt;/changeStamp&gt;   &lt;text&gt;The text of the response.&lt;/text&gt; &lt;/response&gt;</pre> |
| <b>HTTP response headers:</b>       | 200 OK<br>400 Bad request<br>For more information, see <a href="#">HTTP Responses</a> .  |

## Delete Predefined Response

This API deletes a predefined response.

|                               |   |
|-------------------------------|---|
| <b>URL:</b>                   | http://<ServerIP>:<Port>/ccp-webapp/ccp/predefinedResponse/<id>                         |
| <b>HTTP method:</b>           | DELETE  |
| <b>HTTP response headers:</b> | 200 OK<br>400 Bad request<br>For more information, see <a href="#">HTTP Responses</a> . |







## Proxy

The proxy API allows you to update and read proxy server settings. The current system supports a single proxy. If the proxy is enabled, **all feeds** use the proxy.

This API is represented on the SocialMiner user interface in the System Administration panel.



### Note

Only the administrator created during install can use this API.

- [Proxy API Commands, page 115](#)

## Proxy API Commands

This section describes the supported commands for the Proxy API and the parameters for those commands.

### Related Topics

[GET, on page 115](#)

[PUT, on page 116](#)

[Proxy API Parameters, on page 116](#)

## GET

Retrieves the proxy configuration.

|              |   |
|--------------|---|
| URL:         | http://<ServerIP>:<Port>/ccp-webapp/ccp/proxy/default |
| HTTP method: | GET   |

**Example XML response:**

```
<Proxy>
  <host>[ServerIP]</host>
  <port>[Port]</port>
  <exclusions>
    <exclusion>localhost</exclusion>
    <exclusion>*.cisco.com</exclusion>
  </exclusions>
  <enabled>true</enabled>
  <refURL>
    http://[ServerIP]:[Port]/ccp-webapp/ccp/
    proxy/default
  </refURL>
</Proxy>
```

## PUT

Updates the proxy configuration. By default, the configuration is blank and disabled.

|                                     |   |
|-------------------------------------|---|
| <b>URL:</b>                         | http://<ServerIP>:<Port>/ccp-webapp/ccp/proxy/default   |
| <b>HTTP method:</b>                 | PUT   |
| <b>Parameters:</b>                  | See <a href="#">Proxy API Parameters</a> , on page 116.   |
| <b>Example XML request payload:</b> | <pre>&lt;Proxy&gt;   &lt;host&gt;[ServerIP]&lt;/host&gt;   &lt;port&gt;[port]&lt;/port&gt;   &lt;exclusions&gt;     &lt;exclusion&gt;localhost&lt;/exclusion&gt;     &lt;exclusion&gt;*.cisco.com&lt;/exclusion&gt;     &lt;exclusion&gt;161.44.*&lt;/exclusion&gt;     &lt;exclusion&gt;192.168.1.1&lt;/exclusion&gt;   &lt;/exclusions&gt;   &lt;enabled&gt;true&lt;/enabled&gt; &lt;/Proxy&gt;</pre> |

## Proxy API Parameters

Parameters are optional unless otherwise noted.

| Parameter      | Description   | Notes                                  |
|----------------|---|--|
| <b>enabled</b> | True or false. Defines whether or not proxy use is enabled.     | Defaults to false.                     |
| <b>host</b>    | The fully-qualified hostname or IP address of the proxy server. | Required if enabled parameter is true. |
| <b>port</b>    | The http port for the proxy server.                             | Required if enabled parameter is true. |



| Parameter         | Description   | Notes   |
|-------------------|---|---|
| <b>exclusions</b> | A list of host names to exclude from being used by the proxy. | <p>The exclusion list is limited to 255 total characters (not including the &lt;exclusion&gt; tags). There is an additional character per item in the list that acts as a separator.</p> <p>Wildcards can be used. Examples:</p> <ul style="list-style-type: none"><li>• localhost</li><li>• *.cisco.com</li><li>• xxx.yy.*</li></ul> |





# Public URL Prefix for Chat Invitation

This API specifies (sets and retrieves) a publicly accessed SocialMiner server URL as a pre-defined property named "publicPrefix", which by default is not set. We currently support only one public prefix configuration.

- [GET, page 119](#)
- [PUT, page 119](#)

## GET

Get the public prefix settings. If the property is not set, it will return no prefix.

|                       |   |
|-----------------------|---|
| URL:                  | http://<ServerIP>:<Port>/ccp-webapp/ccp/publicprefix/default  |
| HTTP method:          | GET   |
| Example XML response: | <pre>&lt;Publicprefix&gt;   &lt;path&gt;     http[s]://[public_server]:[serverport]     [/optional/path]   &lt;/path&gt;   &lt;refURL&gt;     http://[ServerIP]:[Port]/ccp-webapp/ccp/     publicprefix/default   &lt;/refURL&gt; &lt;/Publicprefix&gt;</pre> See also <a href="#">API Conventions</a> , on page 1. |

## PUT

Updates the proxy configuration with the public prefix.

|              |  |
|--------------|--|
| URL:         | http://<ServerIP>:<Port>/ccp-webapp/ccp/publicprefix/default |
| HTTP method: | PUT  |

|                              |  |
|------------------------------|--|
| <b>Parameters:</b>           | <p><b>prefix</b>(optional)—specifies the public URL for the SocialMiner server and port. If no port number is provided, it defaults to 8000.</p> <p>If an empty prefix is provided (&lt;Prefix&gt;&lt;&lt;prefix&gt; &lt;/prefix&gt;&lt;/Prefix&gt; ), then the prefix will be set to nothing.</p> <p>The value for "public server" cannot be an IP address.</p> |
| <b>Example XML response:</b> | <pre>&lt;Publicprefix&gt;   &lt;path&gt;     http[s]://[public_server]:[serverport]     [/optional/path]&lt;/path&gt; &lt;/Publicprefix&gt;</pre> <p>See also <a href="#">API Conventions</a>, on page 1.</p>  |



## Purge

The Purge API allows you to change settings associated with the database purge feature. Routine database purging is necessary to prevent the file system from filling up.

This API is represented on the SocialMiner user interface in the System Administration panel.



### Note

Only the administrator created during install can use this API.

- [Purge API Commands](#), page 121

## Purge API Commands

This section describes the supported commands for the Purge API and the parameters for those commands.

### Related Topics

[GET \(List\)](#)

[PUT](#)

[Purge API Parameters](#)

## GET (List)

Lists the current purge settings.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/purge |
| <b>HTTP method:</b> | GET   |
| <b>Parameters:</b>  | See <a href="#">Purge API Parameters</a> .    |

**Example XML response:**

```
<PurgeConfig>
  <dataStoreEmergencyPurgeDiskUsage>
    50
  </dataStoreEmergencyPurgeDiskUsage>
  <dataStorePurgeAge>30</dataStorePurgeAge>
  <reportingPurgeAge>550</reportingPurgeAge>
  <reportingPurgeTime>01:00</reportingPurgeTime>
</PurgeConfig>
```

## PUT

Updates the purge settings.

|                                     |   |
|-------------------------------------|---|
| <b>URL:</b>                         | http://<ServerIP>:<Port>/ccp-webapp/ccp/purge   |
| <b>HTTP method:</b>                 | PUT   |
| <b>Parameters:</b>                  | See <a href="#">Purge API Parameters</a> .  |
| <b>Example XML request payload:</b> | <pre>&lt;PurgeConfig&gt;   &lt;dataStoreEmergencyPurgeDiskUsage&gt;     80   &lt;/dataStoreEmergencyPurgeDiskUsage&gt;   &lt;dataStorePurgeAge&gt;30&lt;/dataStorePurgeAge&gt;   &lt;reportingPurgeAge&gt;550&lt;/reportingPurgeAge&gt;   &lt;reportingPurgeTime&gt;01:00&lt;/reportingPurgeTime&gt; &lt;/PurgeConfig&gt;</pre> |
| <b>HTTP response headers:</b>       | A <i>200 OK</i> http header is returned on success.   |

## Purge API Parameters

All parameters are optional.

| Parameter                               | Description   | Notes  |
|---|---|--|
| <b>dataStoreEmergencyPurgeDiskUsage</b> | The percent of disk usage that acts as a purge threshold. When this threshold is reached a purge starts. Social contacts older than dataStorePurgeAge are removed first. If disk usage is still above the threshold for emergency purging, then the purge continues removing social contacts (one day at a time) until the disk usage is below the threshold for emergency purge. | Valid values are 40–90.                            |
| <b>dataStorePurgeAge</b>                | The age of the social contacts that will be purged.   | Must be a whole number between 1–550 (no decimal). |

| Parameter                 | Description  | Notes                            |
|---------------------------|--|----------------------------------|
| <b>reportingPurgeAge</b>  | The time, in 24 hour format (HH:mm), when the purge is to start. | Valid values are 00:00 to 23:59. |
| <b>reportingPurgeTime</b> | The ages of reporting records that will be purged.               | Valid values are 1 to 550.       |







## Push Feed

---

A Push feed (feed type = 7) pushes a new social contact into SocialMiner.

There are two ways to push a social contact into the system:

- using the social contact create (POST) method.
- using a get method.

Both API calls send a message to the Feed subsystem with the new social contact information and the push feed id.

The “pushed” social contact is handled like all other social contacts.

- [Push Feed API Commands, page 125](#)

## Push Feed API Commands

This section describes the supported commands for the Push feed API and the parameters for those commands.

### Related Topics

- [POST, on page 125](#)
- [GET, on page 125](#)

## POST

The social contact create (POST) method ([POST, on page 155](#)) is the preferred method for creating a social contact. Use the payload shown in that API.

## GET

The GET method for push feeds is a simplified alternative to the social contact create (POST) API. It requires no API authentication.

**WARNING:** the GET method is not the preferred API call for creating the social contact. This is because:

- GET does not offer as much information protection as the social contact create (POST) method.
- GET sends no payload body, but contains all parameters for the social contact in the URL. Both the network and the browser could truncate information if the URL is very long. (This would happen if many extensionFields were provided.)

|                                       |   |
|---------------------------------------|---|
| <b>An example of a push feed URL:</b> | http://<ServerIP>:<Port>/ccp-webapp/ccp/pushfeed/<id>?title=Example&author=admin&description=some_example&tags=ccx:sales,ccx:engineering&extensionField_customerID=98765&extensionField_remarks=My_router_is_broken   |
| <b>HTTP method:</b>                   | GET   |
| <b>Parameters:</b>                    | <ul style="list-style-type: none"> <li>• <b>Title</b>—the title of the social contact.</li> <li>• <b>Author</b>—the author of the social contact.</li> <li>• <b>Description</b>—the content of the social contact.</li> <li>• <b>Tags</b>—a comma separated list of tags for this social contact.</li> <li>• <b>ExtensionField_&lt;name&gt;</b>—a custom name and value pair for this social contact.</li> </ul> <p>Tags and the extension field names cannot contain commas or colons.</p> |
| <b>HTTP response headers:</b>         | <p>A 200 response indicates success.</p> <p>See also <a href="#">HTTP Responses</a>.</p>  |



## Reply Template

The Reply template API allows you to add, edit, and delete the name and location of custom reply templates. This API is represented on the SocialMiner user interface in the Templates panel.

- [Reply Template API Commands](#), page 127

## Reply Template API Commands

This section describes the supported commands for the Reply template API and the parameters for those commands.

### Related Topics

- [POST](#), on page 127
- [DELETE](#), on page 128
- [GET \(List\)](#), on page 128
- [GET](#), on page 129
- [PUT](#), on page 129
- [Reply Template API Parameters](#), on page 130

## POST

Creates a new template definition.

|                     |  |
|---------------------|--|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/template/reply  |
| <b>HTTP method:</b> | POST   |
| <b>Parameters:</b>  | See <a href="#">Reply Template API Parameters</a> , on page 130. |

|                                     |  |
|-------------------------------------|--|
| <b>Example XML request payload:</b> | <pre>&lt;Template&gt; &lt;name&gt;My_Template&lt;/name&gt; &lt;templateURL&gt;   http://this.is.my.template.url/template.html &lt;/templateURL&gt; &lt;/Template&gt;</pre> |
| <b>Response:</b>                    | Status: 201 Created  |

## DELETE

Deletes a custom reply template definition.

|                               |  |
|-------------------------------|--|
| <b>URL:</b>                   | http://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/template/reply/<ID Variables> |
| <b>HTTP method:</b>           | DELETE   |
| <b>HTTP response headers:</b> | Status 200: OK   |

## GET (List)

Lists all custom reply templates stored on this system.

|                     |  |
|---------------------|--|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/template/reply/ |
| <b>HTTP method:</b> | GET  |
| <b>Parameters:</b>  | See <a href="#">Reply Template API Parameters</a> , on page 130. |

**Example response:**

```

<Templates>
  <Template>
    <changeStamp>0</changeStamp>
    <name>Cisco Twitter</name>
    <refURL>
      http://[ServerIP]:[Port]/ccp-webapp/ccp/template/
      reply/300
    </refURL>
    <systemDefined>>true</systemDefined>
    <templateURL>
      /gadgets/files/ccp/templates/reply/cisco_twitter.jsp
    </templateURL>
  </Template>
  <Template>
    <changeStamp>0</changeStamp>
    <name>My Test Template</name>
    <refURL>
      http://[ServerIP]:[Port]/ccp-webapp/ccp/
      template/reply/100024
    </refURL>
    <systemDefined>>false</systemDefined>
    <templateURL>
      http://this.is.my.template.url/template.html
    </templateURL>
  </Template>
</Templates>

```

## GET

Gets the details for one reply template.

|                              |   |
|------------------------------|---|
| <b>URL:</b>                  | http://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/template/<br>reply/<ID Variables>  |
| <b>HTTP method:</b>          | GET   |
| <b>Example XML response:</b> | <pre> &lt;Template&gt;   &lt;changeStamp&gt;1&lt;/changeStamp&gt;   &lt;name&gt;Remote Custom Template&lt;/name&gt;   &lt;refURL&gt;     http://[ServerIP]:[Port]/     ccp-webapp/ccp/template/reply/105673   &lt;/refURL&gt;   &lt;systemDefined&gt;&gt;false&lt;/systemDefined&gt;   &lt;templateURL&gt;     http://[ServerIP]:[Port]/ccp-reply/     twitter-reply-gadget.jsp   &lt;/templateURL&gt; &lt;/Template&gt; </pre> |
| <b>Parameters:</b>           | See <a href="#">Reply Template API Parameters</a> , on page 130.  |

## PUT

Updates an existing reply template definition.

|             |  |
|-------------|--|
| <b>URL:</b> | http://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/template/ |
|-------------|--|

|                                     |  |
|-------------------------------------|--|
|                                     | reply/<ID Variables>   |
| <b>HTTP method:</b>                 | PUT  |
| <b>Parameters:</b>                  | See <a href="#">Reply Template API Parameters</a> , on page 130.   |
| <b>Example XML request payload:</b> | <pre> &lt;Template&gt;   &lt;changeStamp&gt;1&lt;/changeStamp&gt;   &lt;name&gt;Remote Custom Template&lt;/name&gt;   &lt;templateURL&gt;     http://[ServerIP]:[Port]/ccp-reply/     twitter-reply-gadget.jsp   &lt;/templateURL&gt; &lt;/Template&gt; </pre> |

## Reply Template API Parameters

Parameters are optional unless otherwise noted.

| Parameter            | Description   | Notes  |
|----------------------|---|--|
| <b>changeStamp</b>   | The change stamp of the reply template record.  | Integer. Defaults to 0.<br>changeStamp is required for PUT (update API).<br>The changeStamp increments by 1 if the update is successful. |
| <b>name</b>          | The template name.  | String.<br>Required for POST.  |
| <b>refURL</b>        | The reference to the reply template.  | Boolean.   |
| <b>systemDefined</b> | True if the template was pre-installed on SocialMiner.  | systemDefined templates cannot be deleted.   |
| <b>templateURL</b>   | The URL of your template. The URL must reference an OpenSocial gadget to be displayed in SocialMiner.<br>Additional information on OpenSocial is available at <a href="http://docs.opensocial.org/display/OS/Home">http://docs.opensocial.org/display/OS/Home</a> . | String.<br>Required for POST.  |



## Reporting Server

The reporting server API returns the reporting Server database connection information. This API is represented on the SocialMiner user interface in the System Administration panel.



### Note

Only the administrator created during install can use this API.

- [Reporting Server API Commands, page 131](#)

## Reporting Server API Commands

This section describes the supported command (GET) for the reporting server API and the parameters for that command.

### GET

Gets the reporting server database connection information.



### Note

Only the administrator created during install can use this API.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/reportingserver |
| <b>HTTP method:</b> | get   |

|                              |   |
|------------------------------|---|
| <b>Example XML response:</b> | <pre> &lt;ReportingServer&gt;   &lt;databaseType&gt;Informix&lt;/databaseType&gt;   &lt;reportingDatabase&gt;[database name]&lt;/reportingDatabase&gt;   &lt;reportingHost&gt;[hostname]&lt;/reportingHost&gt;   &lt;reportingHostIp&gt;[ServerIP]&lt;/reportingHostIp&gt;   &lt;reportingPort&gt;[Port]&lt;/reportingPort&gt;   &lt;reportingServer&gt;[server name]&lt;/reportingServer&gt; &lt;/ReportingServer&gt; </pre> |
| <b>Parameters</b>            | See <a href="#">Reporting Server API Parameters</a> , on page 132.  |

## Reporting Server API Parameters

All parameters are optional.

| Parameter                | Description                                   | Notes               |
|--------------------------|---|---------------------|
| <b>databaseType</b>      | The database server type.                     | Returns "Informix". |
| <b>reportingDatabase</b> | The name of the reporting database.           |                     |
| <b>reportingHost</b>     | The host name of the reporting server.        |                     |
| <b>reportingHostIp</b>   | The IP address of the reporting server.       |                     |
| <b>reportingPort</b>     | The connection port for the reporting server. |                     |
| <b>reportingServer</b>   | The reporting-server informix instance name.  |                     |





## Reporting User

The reporting user API allows you to create the reporting user and set the reporting user password. The reporting user username is *reportinguser* and it is not editable.



### Note

Only the administrator created during install can use this API.

- [Reporting User API Commands, page 133](#)

## Reporting User API Commands

This section describes the supported commands for the reporting user API and the parameters for those commands.

### Related Topics

- [POST, on page 133](#)
- [DELETE, on page 134](#)
- [GET \(List\), on page 134](#)
- [GET, on page 134](#)
- [PUT, on page 135](#)
- [Reporting User API Parameters, on page 135](#)

## POST

Creates the reporting user.



### Note

The username of the reporting user must be *reportinguser* or an error is received.

|      |   |
|------|---|
| URL: | <code>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/reportinguser/</code> |
|------|---|

|                                     |  |
|-------------------------------------|--|
| <b>HTTP method:</b>                 | POST   |
| <b>Parameters:</b>                  | See <a href="#">Reporting User API Parameters</a> , on page 135.   |
| <b>Example XML request payload:</b> | <pre>&lt;ReportingUser&gt;   &lt;username&gt;reportinguser&lt;/username&gt;   &lt;password&gt;password&lt;/password&gt; &lt;/ReportingUser&gt;</pre> |
| <b>HTTP response headers:</b>       | A <i>201 Created</i> http header is returned on success.   |

## DELETE

Deletes the reporting user.

|                               |   |
|-------------------------------|---|
| <b>URL:</b>                   | <a href="http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/reportinguser/&lt;ID Variables&gt;">http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/reportinguser/&lt;ID Variables&gt;</a> |
| <b>HTTP method:</b>           | DELETE  |
| <b>HTTP response headers:</b> | A <i>200 OK</i> http header is returned on success.   |

## GET (List)

Lists all reporting users.



### Note

Currently only a single reporting user is supported.

|                              |  |
|------------------------------|--|
| <b>URL:</b>                  | <a href="http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/reportinguser">http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/reportinguser</a>  |
| <b>HTTP method:</b>          | GET  |
| <b>Parameters</b>            | See <a href="#">Reporting User API Parameters</a> , on page 135.   |
| <b>Example XML response:</b> | <pre>&lt;ReportingUsers&gt;   &lt;ReportingUser&gt;     &lt;refURL&gt;http://[ServerIP]:[Port]/       ccp-webapp/ccp/reportinguser/100001&lt;/refURL&gt;     &lt;username&gt;reportinguser&lt;/username&gt;   &lt;/ReportingUser&gt; &lt;/ReportingUsers&gt;</pre> |

## GET

Gets a single reporting user from the database.

|             |   |
|-------------|---|
| <b>URL:</b> | <a href="http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/reportinguser/&lt;ID Variables&gt;">http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/reportinguser/&lt;ID Variables&gt;</a> |
|-------------|---|

|                          |   |
|--------------------------|---|
| <b>HTTP method:</b>      | GET   |
| <b>Example response:</b> | <pre>&lt;ReportingUser&gt;   &lt;refURL&gt;http://[ServerIP]:[Port]/     ccp-webapp/ccp/reportinguser/100001&lt;/refURL&gt;   &lt;username&gt;reportinguser&lt;/username&gt; &lt;/ReportingUser&gt;</pre> |

## PUT

Updates a reporting user. Currently, only the password can be changed.

|                                     |   |
|-------------------------------------|---|
| <b>URL:</b>                         | http://<ServerIP>:<Port>/ccp-webapp/ccp/reportinguser/<ID Variables>  |
| <b>HTTP method:</b>                 | PUT   |
| <b>Parameters:</b>                  | <ul style="list-style-type: none"> <li>• <b>password:</b> string—the new password for the reporting user.</li> </ul>                                    |
| <b>Example XML request payload:</b> | <pre>&lt;ReportingUser&gt;   &lt;username&gt;reportinguser&lt;/username&gt;   &lt;password&gt;newpassword&lt;/password&gt; &lt;/ReportingUser&gt;</pre> |
| <b>HTTP response headers:</b>       | A <i>200 OK</i> http header is returned on success.   |

## Reporting User API Parameters

| Parameter       | Description                          | Notes                        |
|-----------------|--------------------------------------|------------------------------|
| <b>username</b> | Must be <i>reportinguser</i> .       | String.<br>Required for POST |
| <b>password</b> | The password for the reporting user. | String.<br>Required for POST |





## Serviceability

The serviceability API provides details on the SocialMiner version, service status, and various other information used primarily by SocialMiner developers for debugging issues.



### Note

Only the administrator can use this API.

This API is represented on the SocialMiner user interface in the System Administration panel.

- [Serviceability API Commands, page 137](#)

## Serviceability API Commands

This section describes the supported commands for the serviceability API and the parameters for those commands.

### Related Topics

- [GET, on page 137](#)
- [GET \(List\), on page 138](#)
- [Get \(Performance Information\), on page 141](#)
- [Serviceability API Parameters, on page 143](#)

## GET

Gets the value for a given serviceability attribute. Each piece of information is returned as an XML element named for each category of information.

Any combination of the categories listed below can be retrieved by providing any number of category parameters on the URL.

For example,

```
http://<ServerIP>:<Port>/ccp-webapp/ccp/serviceability?
```

```
category=diskUsage&category=serviceState
```

returns only the *diskUsage* and *serviceStates* categories.

**Note**

With no categories specified, all categories except *systemInfo* are returned. This is because *systemInfo* is very expensive to run as it collects all JMX attributes from all four JVMs in the product (runtime server, cassandra, solr and Tomcat).

To retrieve all categories including *systemInfo*, use the special *all* category:

*http://<ServerIP>:<Port>/ccp-webapp/ccp/serviceability?category=all*

|                              |  |
|------------------------------|--|
| <b>URL:</b>                  | http://<ServerIP>:<Port>/ccp-webapp/ccp/serviceability   |
| <b>HTTP method:</b>          | GET  |
| <b>Parameters:</b>           | See <a href="#">Serviceability API Parameters</a> , on page 143.   |
| <b>Example XML response:</b> | <p>Example response for<br/>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/serviceability/?category=serviceStates</p> <pre> &lt;serviceStates&gt;   &lt;activemqServerState&gt;     SERVER_STATE_IN_SERVICE   &lt;/activemqServerState&gt;   &lt;datastoreServerState&gt;     SERVER_STATE_IN_SERVICE   &lt;/datastoreServerState&gt;   &lt;indexerServerState&gt;     SERVER_STATE_IN_SERVICE   &lt;/indexerServerState&gt;   &lt;runtimeServerState&gt;     SERVER_STATE_IN_SERVICE   &lt;/runtimeServerState&gt; &lt;/serviceStates&gt; </pre> |

## GET (List)

Lists all of the available serviceability attributes and their current values. See [GET](#), on page 137 for details.

|                              |  |
|------------------------------|--|
| <b>URL:</b>                  | http://<ServerIP>:<Port>/ccp-webapp/ccp/serviceability/  |
| <b>HTTP method:</b>          | GET  |
| <b>Example XML response:</b> | <p>This example is an excerpt of the XML response. The full output consists of more than 3 MB of data.</p> <pre> &lt;Serviceability&gt;   &lt;diskUsage&gt;     &lt;activePartitionTotalBytes&gt;       13463810048     &lt;/activePartitionTotalBytes&gt;     &lt;activePartitionUsableBytes&gt;       2352652288     &lt;/activePartitionUsableBytes&gt;     &lt;commonDatastorePartitionTotalBytes&gt;       46835601408     &lt;/commonDatastorePartitionTotalBytes&gt; </pre> |

```

<commonDatastorePartitionUsableBytes>
  37941018624
</commonDatastorePartitionUsableBytes>
<inactivePartitionTotalBytes>
  13463781376
</inactivePartitionTotalBytes>
<inactivePartitionUsableBytes>
  2357100544
</inactivePartitionUsableBytes>
</diskUsage>
...
<facebookAppInfo>
  <facebookAppId>
    facebookAppIdAsString
  </facebookAppId>
</facebookAppInfo>
...
<feedStatuses>
  <FeedStatus>
    <feedRefURL>
      http://[ServerIP]:[Port]/ccp-webapp/ccp/feed/100000
    </feedRefURL>
    <lastFetchCount>20</lastFetchCount>
    <statusDescription>
      NORMAL
    </statusDescription>
  </FeedStatus>
</feedStatuses>
...
<filterStatuses>
  <FilterStatus>
    <timeSinceLastExecution>
      125
    </timeSinceLastExecution>
    <name>TestFilter</name>
    <status>FILTER_EXECUTION_ERROR</name>
    <statusDescription>
      Cannot cast object 'null' with class 'null'
      to class 'int'.
      Try 'java.lang.Integer' instead.
    </statusDescription>
  </FilterStatus>
</filterStatuses>
...
<notifiers>
  <Notifier>
    <connectionStatus>
      STATE_IN_SERVICE
    </connectionStatus>
    <notificationsDropped>
      0
    </notificationsDropped>
    <notificationsFailed>
      0
    </notificationsFailed>
    <notificationsSent>
      10
    </notificationsSent>
    <outQueueDepth>2</outQueueDepth>
    <outQueueWait>1</outQueueWait>
    <type>im</type>
  </Notifier>
</notifiers>
...
<notificationRuleStatuses>
  <NotificationRuleStatus>
    <id>10001</id>
    <name>CCE</name>
    <type>Connection to CCE</type>
    <status>SUCCESS<status>
    <statusReason>
      NOTIFICATION_STATUS_NORMAL

```

```

        <statusReasonDescription>
          The notification status is normal.
        </statusReasonDescription>
        <success>10</success>
        <failure>5</failure>
        <total>15</total>
        <statusChangeTime>
          Sat Oct 30 18:38:41 EDT 2010
        </statusChangeTime>
      </NotificationRuleStatus>
    <NotificationRuleStatus>
      <id>10002</id>
      <name>MyNotification</name>
      <type>http</type>
      <status>FAILURE</status>
      <statusReason>
        NOTIFICATION_STATUS_BAD_CONFIGURATION
      </statusReason>
      <statusReasonDescription>
        Failed to send notification due to
        Bad Configuration.
      </statusReasonDescription>
      <success>0</success>
      <failure>5</failure>
      <total>5</total>
      <statusChangeTime>
        Sat Oct 30 18:38:41 EDT 2010
      </statusChangeTime>
    </NotificationRuleStatus>
  </notificationRuleStatuses>
  ...
  <serviceStates>
    <activemqServerState>
      SERVER_STATE_IN_SERVICE
    </activemqServerState>
    <datastoreServerState>
      SERVER_STATE_IN_SERVICE
    </datastoreServerState>
    <indexerServerState>
      SERVER_STATE_IN_SERVICE
    </indexerServerState>
    <runtimeServerState>
      SERVER_STATE_IN_SERVICE
    </runtimeServerState>
  </serviceStates>

  <systemPerformance>
    <socialContactsPerHour>
      10000
    </socialContactsPerHour>
  </systemPerformance>

  <systemConditions/>
  <version>
    <buildDate>
      Sat Oct 30 18:38:41 EDT 2010
    </buildDate>
    <buildVersion>165</buildVersion>
    <esVersion>0</esVersion>
    <maintenanceVersion>1</maintenanceVersion>
    <majorVersion>8</majorVersion>
    <minorVersion>5</minorVersion>
    <srVersion>0</srVersion>
    <vosActiveVersion>
      8.5.0.97000-93
    </vosActiveVersion>
    <vosInactiveVersion>
      8.5.0.97000-92
    </vosInactiveVersion>
  </version>
  ...
  <eventingInfo>

```



```

<connectionStatus>
  CONNECTED
</connectionStatus>
<dsNfyMsgsRcvd>15</dsNfyMsgsRcvd>
<outQueueDepth>0</outQueueDepth>
<outQueueWait>0</outQueueWait>
<xmppEventsDropped>0</xmppEventsDropped>
<xmppEventsFailed>0</xmppEventsFailed>
<xmppEventsSent>15</xmppEventsSent>
</eventingInfo>
...
</Serviceability>

```

## Get (Performance Information)

Gets information about the performance counters in the system.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | http://<ServerIP>:<Port>/sm-dp/rest/DiagnosticPortal/GetPerformanceInformation  |
| <b>HTTP method:</b> | GET   |
| <b>Elements:</b>    | <p><b>xmlns:</b> XML namespace "dp" defined for Diagnostic Portal specific elements to avoid naming errors.</p> <p><b>Schema Version:</b> This element is defined for tracking the version.</p> <p>The performance API retrieves information in the form of "Property Name" and "Value" for each of the following categories:</p> <p><b>systemInfo:</b></p> <p>The following counters are retrieved for the system services such as Runtime, XMPP, Webapp, Indexer, and Datastore:</p> <ul style="list-style-type: none"> <li>• System/Available Processors</li> <li>• System/Committed Virtual Memory Size</li> <li>• System/Free Physical Memory Size</li> <li>• System/Free Swap Space Size</li> <li>• System/Max File Descriptor Count</li> <li>• System/Open File Descriptor Count</li> <li>• System/Process Cpu Time</li> <li>• System/System Load Average</li> <li>• System/Total Physical Memory Size</li> <li>• System/Total Swap Space Size</li> <li>• Threading/Peak Thread Count</li> <li>• Threading/Thread Count</li> <li>• Threading/Total Started Thread Count</li> </ul> |

The property name is retrieved in the following format:

<service name>/<counter name>

**feedstatuses:** The property name is retrieved in the following format:

Feed Status(feed name)/<counter name>

Information about the following counters is retrieved

- statusDescription
- feedRefURL
- fetchSuccessRate
- lastSuccessfulFetchTimeInSeconds
- lastFetchCount

**notifiers:** The property name is retrieved in the following format:

Notifier(connection name)/<counter name>

Information about the following counters is retrieved

- notificationsSent
- lastFailureCause
- notificationsDropped
- connectionStatus
- notificationsFailed
- outQueueWait
- outQueueDepth

**notificationRuleStatuses:** The property name is retrieved in the following format:

Notification Rule Status(notification name)/<counter name>

Information about the following counters is retrieved

- total
- failure
- status
- type
- Success

**eventingInfo:** The property name is retrieved in the following format:

Eventing Info/<counter name>

Information about the following counters is retrieved

- dsNfyMsgsRcvd
- connectionStatus

- xmppEventsDropped
- xmppEventsSent
- outQueueWait
- xmppEventsFailed
- outQueueDepth

For information about the elements, see [Serviceability API Parameters](#), on page 143.

**Example  
XML  
response:**

```
<dp:GetPerformanceInformationReply
xmlns:dp="http://www.cisco.com/vtg/diagnosticportal" ReturnCode="0">
  <dp:Schema Version="1.0"/>
  <dp:PerformanceInformation>
    <dp:PropertyList>
      <dp:Property Name="Runtime/System/Total Physical Memory Size"
Value="8508125184"/>
      <dp:Property Name="Runtime/System/System Load Average" Value="0.82"/>
      <dp:Property Name="Runtime/Threading/Thread Count" Value="581"/>
      <dp:Property Name="Runtime/System/Process Cpu Time" Value="2695750000000"/>

      <dp:Property Name="Runtime/Threading/Total Started Thread Count"
Value="8235"/>
      <dp:Property Name="Runtime/System/Free Swap Space Size" Value="2113822720"/>

      <dp:Property Name="Runtime/System/Max File Descriptor Count" Value="64000"/>

      <dp:Property Name="Runtime/System/Committed Virtual Memory Size"
Value="2046943232"/>
      <dp:Property Name="Runtime/System/Available Processors" Value="2"/>
      <dp:Property Name="Runtime/Threading/Peak Thread Count" Value="618"/>
      <dp:Property Name="Runtime/System/Free Physical Memory Size"
Value="1680343040"/>
      <dp:Property Name="Runtime/System/Open File Descriptor Count" Value="245"/>

      <dp:Property Name="Runtime/System/Total Swap Space Size" Value="2113822720"/>

      <dp:Property Name="Feed Status(Feed Name)/statusDescription"
Value="NETWORK_NOT_REACHABLE"/>
      <dp:Property Name="Eventing Info/dsNfyMsgsRcvd" Value="220"/>
      <dp:Property Name="Notifier(Connection Name)/notificationsSent" Value="11"/>

      <dp:Property Name="Notification Rule Status(Notification Name)/total"
Value="1"/>
    /dp:PropertyList>
  /dp:PerformanceInformation>
</dp:GetPerformanceInformationReply>
```

## Serviceability API Parameters

Parameters are optional unless otherwise noted.

| Parameter            | Description   | Notes   |
|----------------------|---|---|
| <b>apistats</b>      | Timing statistics on each API type exposed by the system.         | All time units are in milliseconds.<br>The interval for “lastInterval” stats is 30 seconds. |
| <b>configuration</b> | An XML representation of all configuration objects in the system. | This is the same XML that the API produces and consumes.                                    |

| Parameter        | Description   | Notes  |
|------------------|---|--|
| <b>diskusage</b> | Reports on the total size in bytes and remaining usable bytes for each partition on the disk. | <p>In some small deployments, there will be no reportingDatabase partition because it is the same as the commonDatastore partition.</p> <p>In large deployments, the partitions reported for both total and usable size will be:</p> <ul style="list-style-type: none"><li>• activePartition—main OS partition. Contains config database. Mounted on <i>/</i>.</li><li>• inactivePartition—inactive side used to upgrade. Will be activePartition after a switch. Mounted on <i>/partB</i>.</li><li>• commonDatastorePartition—social contact datastore. Both Cassandra and Solr data files. Mounted on <i>/common</i>.</li><li>• reportingDatabasePartition—contains the reporting database. Mounted on <i>/spare</i>. On small deployments this is sometimes a symlink into <i>/common</i>. In those cases, this partition will be omitted from the results.</li></ul> |

| Parameter              | Description  | Notes  |
|------------------------|--|--|
| <b>eventingInfo</b>    | Statistics about the Eventing Subsystem which publishes social contact state change events to XMPP clients using the embedded XMPP server. | <ul style="list-style-type: none"> <li>• <b>dsNfyMsgsRcvd</b>: the number of social contact state change messages received from the data store.</li> <li>• <b>xmppEventsSent</b>: the number of xmpp events successfully sent by this notifier.</li> <li>• <b>xmppEventFailed</b>: the number of failed xmpp notification attempts.</li> <li>• <b>xmppEventDropped</b>: the number of xmpp events that were dropped because the output queue was full.</li> <li>• <b>outQueueDepth</b>: the number of items in the output queue.</li> <li>• <b>outQueueWait</b>: the average amount of time between when an xmpp event is queued and when it is sent.</li> <li>• <b>connectionStatus</b>: one of the following states: <ul style="list-style-type: none"> <li>◦ <i>CONNECTED</i>: successfully connected to the embedded XMPP server.</li> <li>◦ <i>DISCONNECTED</i>: the notifier is unable to connect to the configured server.</li> <li>◦ <i>DISABLED</i>: the notifier is not enabled.</li> <li>◦ <i>BAD_CONFIGURATION</i>: the notifier is not configured correctly and is unable to attempt to make a connection.</li> </ul> </li> </ul> |
| <b>facebookAppInfo</b> | SocialMiner's Facebook Application ID.   |  |

| Parameter             | Description  | Notes   |
|-----------------------|--|---|
| <b>feedstatuses</b>   | Any number of FeedStatus elements containing the last known status of every feed in the system that is currently being fetched because at least one campaign has that feed configured in it.<br><br>See <a href="#">Feed statusDescription Values, on page 153</a> . | FeedStatus can include this information: <ul style="list-style-type: none"> <li>• <b>FETCH_SUCCESS_RATE</b>: calculated based on the last 10 fetches:<br/> <math>\text{FetchSuccessRate} = ((\text{FetchSuccess} / (\text{FetchSuccess} + \text{FetchFailure})) / 100)</math>.<br/> Applies to these classes: ccp-feeds and ccp-webapp.</li> <li>• <b>LAST_GOOD_FETCH_TIME</b>: applies to classes ccp-feeds and ccp-webapp.</li> </ul>   |
| <b>filterStatuses</b> | FilterStatus elements containing the last known status of every filter in the system.  | <ul style="list-style-type: none"> <li>• <b>name</b>: filter name</li> <li>• <b>status</b>: NORMAL or FILTER_EXECUTION_ERROR (for script filter). As the runtime does not track whether a filter is in a campaign, that information is not available in filter status.</li> <li>• <b>status description</b>: a detailed error message in the case of a script filter execution error. Otherwise, normal.</li> <li>• <b>timeSinceLastExecution</b>: the time in seconds since the last execution of the script filter. For all other filter types and for a script filter that has never been executed since runtime started, returns -1.</li> </ul> |
| <b>hardware</b>       | The status of the hardware SMALL, LARGE, UNSUPPORTED.  | diskOneSize: size of disk 1.<br>diskTwoSize: size of disk 2.<br>diskOneStar: start byte of disk 1.<br>diskTwoStart: start byte of disk 2.<br>memory: total physical memory.<br>numberOfCpus: total number of cpus(cores).<br>type: SMALL, LARGE, or UNSUPPORTED. Is determined by looking at the values listed above and comparing them to known ovf values.  |

| Parameter | Description   | Notes   |
|-----------|---|---|
| jvmStats  | Statistics about each running Java Virtual Machine (JVM) on the server. These are the same as the serviceStates category. | <p>The stats for each JVM are:</p> <ul style="list-style-type: none"><li>• <b>connectionName</b>: the name and JMX URL (only accessible from the local machine) of the JVM.</li><li>• <b>cpuPercentageSamples</b>: samples of the percentage of CPU used by this JVM. Each sample is a 30 second interval. Five days of samples are kept.</li><li>• <b>heapSamples</b>: samples of the total heap usage in MB. Each sample is a 2 minute interval. One week of samples are kept.</li><li>• <b>heapSlope</b>: the slope of the graph of the heapSamples. After a long period of uptime this should be very close to 0 (flat slope) indicating no heap memory leakage over time. Over short periods the slope is not useful because of the volatility of the heap usage.</li><li>• <b>uptime</b>: elapsed time since this JVM started in milliseconds.</li><li>• <b>uptimeString</b>: a human readable version of uptime.</li></ul> |

| Parameter        | Description  | Notes  |
|------------------|--|--|
| <b>notifiers</b> | The status of the notification services for this server. | <p><b>For "email", "im" and "http" notification types, associated data are</b></p> <ul style="list-style-type: none"> <li>• <b>type</b>: the type of notification sent by this notifier.</li> <li>• <b>notificationsSent</b>: the number of notifications successfully sent by this notifier.</li> <li>• <b>notificationsFailed</b>: the number of failed notifications.</li> <li>• <b>notificationsDropped</b>: the number of notifications that were dropped because the output queue was full.</li> <li>• <b>outQueueDepth</b>: the number of items in the output queue.</li> <li>• <b>outQueueWait</b>: the average amount of time in milliseconds between when a notification request is queued and when it is sent.</li> </ul> <p>• <b>ConnectionStatus</b>: Can be one of</p> <ul style="list-style-type: none"> <li>◦ <i>CONNECTED</i>: the notifier successfully connected to the configured server.</li> <li>◦ <i>DISCONNECTED</i>: the notifier is unable to connect to the configured server.</li> <li>◦ <i>DISABLED</i>: the notifier is not enabled.</li> <li>◦ <i>BAD_CONFIGURATION</i>: the notifier is not configured correctly and is unable to attempt to make a connection.</li> </ul> |



| Parameter | Description | Notes   |
|-----------|-------------|---|
| notifiers |             | <p><b>For "connection to cce" notification type, associated data are</b></p> <ul style="list-style-type: none"> <li>• type: connection to cce.</li> <li>• notificationsSent: the number of tasks successfully allocated to an agent by cce.</li> <li>• notificationsFailed: the number of tasks failed to find an agent.</li> <li>• notificationsDropped: the number of tasks that were dropped because there was no connection to cce.</li> <li>• outQueueDepth: the number of tasks currently being processed by cce.</li> <li>• outQueueWait: 0.</li> </ul><br><ul style="list-style-type: none"> <li>• <b>ConnectionStatus:</b> Can be one of <ul style="list-style-type: none"> <li>◦ <i>NOT_ESTABLISHED</i>: CCE configuration for multichannel routing is disabled.</li> <li>◦ <i>LISTENING</i>: listening for incoming connection from CCE media routing PG.</li> <li>◦ <i>ESTABLISHING</i>: establishing connection from CCE media routing PG.</li> <li>◦ <i>ESTABLISHED_ROUTING_ENABLED</i>: established connection with CCE media routing PG.</li> </ul> </li> </ul> |

| Parameter                            | Description                                       | Notes  |
|--------------------------------------|---|--|
| <b>notificationRule<br/>Statuses</b> | Status and statistics for each notification rule. | <ul style="list-style-type: none"><li>• id: the id of the notification rule.</li><li>• name: the name of the notification rule.</li><li>• type: the type of notification rule.</li><li>• status: an explanation of the status for the most recent notification sent.<ul style="list-style-type: none"><li>◦ success: the number of notifications that were successfully sent.</li><li>◦ failure: the number of notifications that failed to be sent.</li><li>◦ total: The total number of notifications processed.</li></ul></li></ul> |

| Parameter | Description | Notes  |
|-----------|-------------|--|
|           |             | <ul style="list-style-type: none"> <li>• <b>statusReason</b>: Reason code for the current status. <ul style="list-style-type: none"> <li>◦ <b>NOTIFICATION_STATUS_UNEXPECTED_ERROR</b></li> <li>◦ <b>NOTIFICATION_STATUS_NORMAL</b></li> <li>◦ <b>NOTIFICATION_STATUS_BAD_CONFIGURATION</b></li> <li>◦ <b>NOTIFICATION_STATUS_CONNECTION_PROBLEM</b></li> <li>◦ <b>NOTIFICATION_STATUS_AUTHENTICATION_FAILED</b></li> <li>◦ <b>NOTIFICATION_STATUS_CCE_ROUTING_PROBLEM</b></li> <li>◦ <b>NOTIFICATION_STATUS_NO_CONNECTION_TO_CCE</b></li> <li>◦ <b>NOTIFICATION_STATUS_CCE_MESSAGE_QUEUE_FULL</b></li> <li>◦ <b>NOTIFICATION_STATUS_RATE_LIMITED</b></li> </ul> </li> </ul> |

| Parameter               | Description   | Notes  |
|-------------------------|---|--|
|                         |   | <ul style="list-style-type: none"> <li>• statusReasonDescription: an explanation for the current statusReason code.</li> <li>• statusChangeTime: time when the status was last changed.</li> </ul>   |
| <b>serviceStates</b>    | Get the state of services.  | <p>Values for serverState can be:</p> <ul style="list-style-type: none"> <li>• SERVER_STATE_API_INIT: the Serviceability API is initializing. This is the state when the API is first started.</li> <li>• SERVER_STATE_UNREACHABLE: the API can't connect to the Runtime, Datastore, or Indexer Server to check the state either because the service is down, stopped, or because of other errors.</li> <li>• SERVER_STATE_IN_SERVICE: the Runtime, Datastore, or Indexer server is in service.</li> <li>• SERVER_STATE_PARTIAL_SERVICE: the Runtime, Datastore, or Indexer server is waiting for another component or sub-component to start or recover from an error. No new social contacts are returned when the service is in partial service.</li> </ul> |
| <b>sessionInfo</b>      | A list of current sessions (ip address:username combination).                                 | This is used to indicate active sessions and calculate approximate number of logged in users.  |
| <b>systemInfo</b>       | Get all available system parameters.  | <b>Important!</b> Accessing this category forces the system to dump all system parameters into a large XML file. System performance is greatly inhibited while the snapshot is created.  |
| <b>systemConditions</b> | Get a list of SystemCondition elements describing persistent states reported by the services. |  |

| Parameter                | Description   | Notes  |
|--------------------------|---|--|
| <b>systemPerformance</b> | <b>socialContactsPerHour:</b> This value will be displayed in the Admin gadget along with an icon to show the user if they are in the safe (< 8,000), warning (between 8,000 and 10,000), or critical range (> 10,000). | The socialContactsPerHour comes from the ReportingIntervalStatsMbean. It retrieves the number of social contacts every 15 minutes and then to get the social contacts per hour it averages the last 4 intervals. |
| <b>version</b>           | Application version information.  | Shows the currently installed version of SocialMiner and the prior version of SocialMiner.   |

## Feed statusDescription Values

Values for a feed statusDescription can be:

- **AUTHENTICATION\_FAILED:** an authenticated feed failed because of incorrect credentials.
- **DATASTORE\_ERROR:** error attempting to write the social contacts to the datastore.
- **FACEBOOK\_AUTHORIZATION\_FAILED:** Authorization was denied when SocialMiner attempted to pull contacts from Facebook.
- **FACEBOOK\_PARSE\_ERROR:** The contacts that the feed pulled from Facebook could not be parsed. This may be due to an unknown change in the Facebook API.
- **NETWORK\_ERROR:** unhandled network error.
- **NETWORK\_NOT\_REACHABLE:** could not connect to the remote server.
- **NETWORK\_TIMEOUT:** the remote server was reachable but did not respond to the request in a timely manner.
- **NORMAL:** the feed is operating normally.
- **RATE\_LIMIT\_REACHED:** A Twitter account, search, or stream feed has exceeded the number of requests that Twitter allows it to make over a given period of time. The status indicates when Twitter will allow SocialMiner to begin pulling contacts again.
- **SCHEDULED:** the feed has been scheduled but has not yet been executed. Feeds are in this state for a very short period of time and then either go to NORMAL or an error state.
- **TWITTER\_STREAM\_INTERRUPTED:** the thread that runs to the Twitter stream client was interrupted.
- **TWITTER\_STREAM\_CONNECTED:** the Twitter stream feed has connected to Twitter, but has not yet received any contacts.
- **TWITTER\_STREAM\_MALFORMED\_URL:** an exception occurred when connecting to Twitter.
- **TWITTER\_STREAM\_CONNECT\_ERROR:** there was an error while connecting to the Twitter stream. Requests to Twitter are automatically slowed exponentially to reestablish the connection.
- **TWITTER\_STREAM\_READ\_ERROR:** there was an error while reading to the Twitter stream. Requests to Twitter are automatically slowed linearly.

- **TWITTER\_STREAM\_STREAM\_DISCONNECTED**: a Twitter Stream Feed is subscribed, but not connected to Twitter.
- **TWITTER\_STREAM\_EOF**: Twitter sent an End Of File to close the stream.
- **TWITTER\_STREAM\_NO\_CLIENT**: the Twitter feed checked to see if there were any contacts but no stream client was found for this feed.
- **UNKNOWN\_ERROR**: an error occurred that does not have a specific exception.
- **UNSUPPORTED\_FEED\_CONTENT**: the content retrieved from the feed is not in a format that SocialMiner supports.



## Social Contact

Social contacts are the individual results obtained by campaigns. The social contact API allows you to get and update an individual social contact.

The status of a social contact is global across all campaigns.

- [Social Contact API Commands](#), page 155

## Social Contact API Commands

This section describes the supported commands for the Social Contact API and the parameters for those commands.

### POST

Creates a social contact for a [Push Feed](#).

Before you do this, you must:

- Create a type 7 push feed and [POST](#) it.
- Confirm that the post returned a 201 (success) response code, then look in the location field of the http response for the reference URL (refURL) of the feed just created.
- Add the chat feed to a campaign. You can create a new campaign and then use the PUT API to add the feed to it or to any existing campaign.

|                     |  |
|---------------------|--|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/socialcontact/           |
| <b>HTTP method:</b> | POST   |
| <b>Parameters:</b>  | See <a href="#">Social Contact API Parameters</a> , on page 164. |

**Example XML  
request payload:**

```
<SocialContact>
  <feedRefURL>http://[ServerIP]:[Port]/
    ccp-webapp/ccp/feed/(id)</feedRefURL>
  <title>social contact title</title>
  <publishedDate>social contact date of publish</publishedDate>
  <author>Customer_Name</author>
  <isInvited>true</isInvited>
  <description type="html">
    This is the content of the social contact.
    Perhaps it was a tweet or a blog post.
  </description>
  <tags>
    <tag>tag1</tag>
    <tag>tag2</tag>
  </tags>
  <extensionFields>
    <extensionField>
      <name>accountNumber</name>
      <value>6722392</value>
    </extensionField>
    <extensionField>
      <name>remarks</name>
      <value>My CRS-3 is not cooling enough</value>
    </extensionField>
  </extensionFields>
  <isSoftLocked>false</isSoftLocked>
</SocialContact>
```

**Note** During creation of a social contact, the following fields are restricted (they can only be set by the system):

- **inviteStatus:** is the status of chat invitations sent (if any) from this social contact. The default is NONE.
- **shortURLIds:** is a list of short URL Ids which were generated for this social contact. This list is updated when a short URL is created for a social contact.

You can set the QUEUED status from the API. However, currently it is only set internally by SocialMiner. SocialMiner sets the social contact status to QUEUED when a social contact is to be or has been routed to some external entity.

**HTTP response:**

A *201 Created* http header is returned on success along with a URL to the newly created social contact.

## GET

Gets status and detail for the specified social contact.

The <objectId> attribute required for this command is found in campaign results, in the feed/entry/link rel="socialcontact" element. For example: <link rel="socialcontact" href="http://<ServerIP>:<Port>/ccp-webapp/ccp/socialcontact/22E00F5310000129460A1EB40A568DDE" />

|                     |  |
|---------------------|--|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/socialcontact/<ID Variables> |
| <b>HTTP method:</b> | GET  |



**Example responses:**

Results are returned as XML.

```

<SocialContact>
  <refURL>
    http://[ServerIP]:[Port]/ccp-webapp/
    ccp/socialcontact/
    F9B43F9B100001282E3C18380A568DDD
  </refURL>
  <statusTimestamp>1276008213</statusTimestamp>
  <status>Reserved</status>
  <statusUserId>admin</statusUserId>
  <tags>
    <tag>tag1</tag>
    <tag>tag2</tag>
  </tags>
  <extensionFields>
    <extensionField>
      <name>accountNumber</name>
      <value>13131313</value>
    </extensionField>
  </extensionFields>
  <sourceType>chat</sourceType>
  <isInvited>>false</isInvited>
  <inviteStatus>sent</inviteStatus>
  <shortUrlIds>
    <shortUrlId>
      6DCFBB7A10000139000076C00A568DDB
    </shortUrlId>
  </shortUrlIds>
  <transcriptRefURL>
    http://[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/
    F9B43F9B100001282E3C18380A568DDD/transcript
  </transcriptRefURL>
  <statusUserId>myname</statusUserId>
  <replyTemplateRefURL>
    http://[ServerIP]:[Port]/ccp-webapp/ccp/
    template/reply/105678
  </replyTemplateRefURL>
  <replyTemplateURL>
    http://[ServerIP]:[Port]/gadgets/files/ccp/templates/
    reply/cisco_twitter.jsp
  </replyTemplateURL>
  <id>F9B43F9B100001282E3C18380A568DDD</id>
  <isSoftLocked>>false</isSoftLocked>
</SocialContact>

<SocialContact>
  <author>AlmaDao1@twitter.com</author>
  <createdDate>1317829257152</createdDate>
  <description>
    Top 50 Companies With Best CSR Announced October 5 on
    <em></em>Boston</em> College Webinar | 3BL Media: <a
    href="http://t.co/VXbzViaj">http://t.co/VXbzViaj</a>
    via @<a class="&quot;>
    href="http://twitter.com/AddThis">
    &AddThis</a>
  </description>
  <extensionFields/>
  <id>D4BEA7BC100001320003D80A0A568DF8</id>
  <link>
    http://twitter.com/AlmaDao1/statuses/121610791286878209
  </link>
  <publishedDate>1317829246000</publishedDate>
  <refURL>
    https://[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/
    D4BEA7BC100001320003D80A0A568DF8
  </refURL>
  <replyTemplateRefURL>
    https://[ServerIP]:[Port]/ccp-webapp/ccp/template/reply/
    103184

```

```

</replyTemplateRefURL>
<replyTemplateURL>http://test.com</replyTemplateURL>
<replyToId></replyToId>
<status>unread</status>
<statusTimestamp>1317829257148</statusTimestamp>
<statusUserId></statusUserId>
<tags/>
<title>
  Top 50 Companies With Best CSR
  Announced October 5 on Boston
  College Webinar | 3BL Media:
  http://t.co/VXbzViaj via @AddThis
</title>
</SocialContact>

```

**Note** If *statusUserId* is blank and the status is *unread*, then this social contact has never had a status change.

If the social contact is associated with a feed that supports reply templates, then the *replyTemplateRefURL* and *replyTemplateURL* fields are included. These fields cannot be changed by the social contact API.

You can set the QUEUED status from the API. However, currently it is only set internally by SocialMiner. SocialMiner sets the social contact status to QUEUED when a social contact is to be or has been routed to some external entity.

## PUT (update)

Updates the status or tags of an existing social contact.

You can also add, edit, or remove tags using the update command.

|                     |  |
|---------------------|--|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/socialcontact/<ID Variables> |
| <b>HTTP method:</b> | PUT  |
| <b>Parameters:</b>  | See <a href="#">Social Contact API Parameters</a> , on page 164.     |

**Example XML request payload:**

```
<SocialContact>
  <statusTimestamp>1276008213</statusTimestamp>
  <status>Reserved</status>
  <statusUserId>admin</statusUserId>
  <tags>
    <tag>cool</tag>
    <tag>fresh</tag>
  </tags>
  <replyTemplateRefURL>http://[ServerIP]:[Port]/ccp-webapp/ccp/
    template/reply/105678</replyTemplateRefURL>
  <replyTemplateURL>http://[ServerIP]:[Port]/gadgets/files/ccp/
    templates/reply/cisco_twitter.jsp</replyTemplateURL>
  <isSoftLocked>>false</isSoftLocked>
</SocialContact>
```

**Note** If the social contact is associated with a feed that supports reply templates, then the *replyTemplateRefURL* and *replyTemplateURL* fields are included. These fields can not be changed by the social contact API. The fields *draftResponse* and *draftAction* are saved only with a social contact update that transitions to state **draft**. If the social contact transitions to the handled state, then the *draftResponse* and *draftAction* are cleared.

Updates to following fields are restricted (updates can only be made by the system):

- *isInvited*: this field is set to true if the chat contact was created as a result of a chat invitation. This field is set by the system when the chat contact is created. It is set to false by default for all other social contacts.
- *shortURLIds*: is a list of short URL Ids which were generated for this social contact. This list is updated when a short URL is created for a social contact.

You can set the QUEUED status from the API. However, currently it is only set internally by SocialMiner. SocialMiner sets the social contact status to QUEUED when a social contact is to be or has been routed to some external entity.

**Example XML response:**

```
<SocialContact>
  <refURL>http://[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/
    22E00F5310000129460A1EB40A568DDE</refURL>
  <status>reserved</status>
  <statusTimestamp>1276190792688</statusTimestamp>
  <tags>
    <tag>cool</tag>
    <tag>fresh</tag>
  </tags>
  <statusUserId>admin</statusUserId>
</SocialContact>
```

## GET (Chat Transcript)

Retrieves the transcript for a chat contact (is only valid for chat feeds). This API returns an error if the social contact does not exist, the socialContact sourceType does not equal chat, or the chat transcript is not found.

**Note**

The id is the social contact ID of the chat contact.

|                               |   |
|-------------------------------|---|
| <b>URL:</b>                   | http://<ServerIP>:<Port>/ccp-webapp/ccp/socialcontact/<ID Variables>/transcript   |
| <b>HTTP method:</b>           | GET   |
| <b>Parameters:</b>            | See <a href="#">Social Contact API Parameters</a> , on page 164.  |
| <b>Example XML response :</b> | <pre> &lt;ChatTranscript&gt;   &lt;id&gt;4C01DF3B10000145000010F60A568DD9&lt;/id&gt;   &lt;refURL&gt;http://[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/     4C01DF3B10000145000010F60A568DD9/transcript&lt;/refURL&gt;   &lt;endDate&gt;1326918175931&lt;/endDate&gt;   &lt;participants&gt;     &lt;participant&gt;       &lt;agentName/&gt;       &lt;nickName&gt;Tetyana&lt;/nickName&gt;     &lt;/participant&gt;     &lt;participant&gt;       &lt;agentName&gt;admin&lt;/agentName&gt;       &lt;nickName&gt;Agent&lt;/nickName&gt;     &lt;/participant&gt;   &lt;/participants&gt;   &lt;startDate&gt;1326918153649&lt;/startDate&gt;   &lt;transcript&gt;     &lt;chat&gt;       &lt;time&gt;1326918172743&lt;/time&gt;       &lt;name&gt;Agent&lt;/name&gt;       &lt;msg&gt;hello&lt;/msg&gt;     &lt;/chat&gt;     &lt;chat&gt;       &lt;time&gt;1326918173170&lt;/time&gt;       &lt;name&gt;Agent&lt;/name&gt;       &lt;msg&gt;How can I help you?&lt;/msg&gt;     &lt;/chat&gt;   &lt;/transcript&gt; &lt;/ChatTranscript&gt; </pre> |

## GET (Search)

Use GET (search) to search for social contacts. GET (search) is based on a Solr search. Wildcard-based searches using "?" for a single-character and "\*" for multiple-characters are supported for the fields specified.

You can perform a default search or a field-specific search.

A default search is a freeform search. To perform a default search, add the term you want to search for after "q=" in the GET URL. For example, to search for jsmith, use the following:

http://<ServerIP>:<Port>/ccp-webapp/ccp/search/contacts?q=jsmith

A default search searches the following fields:

- sc.author
- sc.title
- sc.tags
- sc.description
- sc.socialContactStatus
- sc.sourceType
- sc.isSoftLocked

- chat.agentName
- chat.agentNickname
- chat.transcript

The sc.title, sc.description, and chat.transcript field searches are not case-sensitive. To search in all other fields, the search term you enter must be an exact (case-sensitive) match.

**Note**

The chat.agentName is always saved in the database in lower case text, no matter how an agent signs in. For example, if an agent signs in as JSmith, you must search for jsmith to find the contact.

To perform a field-specific search, specify the field in which you want to search. The pairs of field names and values in a search query use the following syntax: *Solr\_field\_name:value*. You can perform a field-specific search on the following fields:

- sc.author: The person who created the social contact (chat request, Tweet, Facebook post).
- sc.link: The link to the social contact.
- sc.publishedDate: The date the social contact was published.
- sc.createdDate: The date the social contact was created.
- sc.socialContactStatus: The status of the social contact (unread, reserved, handled, discarded, draft, or queued).
- sc.socialContactStatusDate: The date that the status of the social contact last changed.
- sc.tags: The tags applied to the social contact. Tags can be applied automatically by the system when the contact enters a feed or manually by a user.
- sc.sourceType: The feed type to which the social contact belongs.
- chat.agentNickname: Nickname for the agent in the chat room.
- chat.agentName: Login username for the agent in the chat room.
- de.id: The social contact ID. This ID is upper-case. The field is case-sensitive.

The search name and value pairs can be joined in logical expressions by AND or OR. The search terms should be encoded if they contain Solr special characters. (For more details, see <http://wiki.apache.org/solr/SolrQuerySyntax>.)

**Note**

The following limitations apply to the search function:

- Field-based searches search only the specified field for the given term or terms (multiple terms are enclosed in double quotes).

For example, to search for contacts authored by John Smith, you can search for `sc.author:John*`, `sc.author:*Smith`, or `sc.author:"John Smith"`.

- You can do wildcard-based searches of these fields.

```
sc.author
sc.tags
chat.agentName
chat.agentNickname
```

For example, to find social contacts with authors `chatAuthorA`, `chatAuthorB`, or `chatAuthorC`, you can search for `chatAuthor*`.

- There is a 32-character limit to the word length (it can only search for words up to 32-characters long).

Search results can contain from 0 to 200 entries. ChatTranscript can be found inside each entry that corresponds to the social contact of type chat.

|                       |  |
|-----------------------|--|
| <b>URL:</b>           | <code>http://ServerIP:8080/ccp-webapp/ccp/search/contacts</code>   |
| <b>HTTP method:</b>   | GET  |
| <b>Output format:</b> | ATOM   |
| <b>Parameters:</b>    | <ul style="list-style-type: none"> <li>• <i>q</i> (required). Query parameter.</li> <li>• <i>count</i> (optional.) Defines how many results to return (default = 50, max = 200).</li> <li>• <i>startIndex</i> (optional). Identifies the index of the first search result. Because we are using Solr for querying, the first index should be 0 (default = 0).</li> </ul> |

|                               |  |
|-------------------------------|--|
| <b>Example query request:</b> | <pre>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/ search/contacts?q=chat.agentName:admin%20AND%20sc. socialContactStatus:handled</pre> <p>As illustrated in the previous example, the search request must be URL encoded in order to work (for example, spaces in the text are represented by "%20").</p>  |
| <b>Example XML response:</b>  | <pre>&lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;feed xmlns="http://www.w3.org/2005/Atom" xmlns:dc= "http://purl.org/dc/elements/1.1/" xmlns:ccp="http://www.cisco.com/ccbu/ccp/xml/socialcontact/1.0/" xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/"&gt;  &lt;title&gt;Results of Search&lt;/title&gt; &lt;link rel="self" href=" http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/ search/contacts?q=chat.agentName:admin%20AND%20sc.socialContactStatus: handled&amp;amp;count=50&amp;amp;startIndex=0" /&gt; &lt;subtitle&gt;This feed has been created by Cisco SocialMiner&lt;/subtitle&gt; &lt;id&gt;http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/search/contacts&lt;/id&gt; &lt;updated&gt;2014-01-14T18:46:27Z&lt;/updated&gt; &lt;dc:date&gt;2014-01-14T18:46:27Z&lt;/dc:date&gt; &lt;opensearch:itemsPerPage&gt;1&lt;/opensearch:itemsPerPage&gt; &lt;opensearch:totalResults&gt;1&lt;/opensearch:totalResults&gt; &lt;opensearch:startIndex&gt;1&lt;/opensearch:startIndex&gt; &lt;opensearch:Query role="request" searchTerms="chat.nickname:Tetyana"/&gt; &lt;opensearch:link rel="search" type="application/opensearchdescription+xml"  href="http://www.cisco.com/opensearch-description.xml" /&gt; &lt;entry&gt; &lt;title&gt;title14&lt;/title&gt; &lt;link rel="alternate" href="http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/ socialcontact/91E6B21910000143000032B10A568DD9" /&gt; &lt;link rel="socialcontact" href="http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/ socialcontact/91E6B21910000143000032B10A568DD9" /&gt; &lt;author&gt; &lt;name&gt;author14&lt;/name&gt; &lt;/author&gt; &lt;id&gt;http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp/socialcontact/ 91E6B21910000143000032B10A568DD9&lt;/id&gt; &lt;updated&gt;2014-01-14T17:57:28Z&lt;/updated&gt; &lt;published&gt;2014-01-14T17:57:28Z&lt;/published&gt; &lt;content type="application/xml"&gt; &lt;ChatTranscript xmlns=""&gt; &lt;id&gt;F27C35DD10000134000007160A568DDD&lt;/id&gt; &lt;endDate&gt;1326918175931&lt;/endDate&gt; &lt;participants&gt; &lt;participant&gt; &lt;agentName&gt;admin&lt;/agentName&gt; &lt;nickName&gt;Agent&lt;/nickName&gt; &lt;/participant&gt; &lt;/participants&gt; &lt;startDate&gt;1326918153649&lt;/startDate&gt; &lt;transcript&gt; &lt;chat&gt; &lt;time&gt;1326918172743&lt;/time&gt; &lt;name&gt;Agent&lt;/name&gt; &lt;msg&gt;Hello, how can I help you?&lt;/msg&gt; &lt;/chat&gt; &lt;chat&gt; &lt;time&gt;1326918173170&lt;/time&gt; &lt;name&gt;author14&lt;/name&gt; &lt;msg&gt;Never mind. I'm all set, thanks.&lt;/msg&gt; &lt;/chat&gt; &lt;/transcript&gt; &lt;/ChatTranscript&gt; &lt;/content&gt; &lt;dc:creator&gt;author14&lt;/dc:creator&gt; &lt;dc:date&gt;2012-01-18T20:22:33Z&lt;/dc:date&gt; &lt;ccp:scstatustimestamp&gt;1326918177731&lt;/ccp:scstatustimestamp&gt; &lt;ccp:scstatus&gt;handled&lt;/ccp:scstatus&gt;</pre> |

```

<ccp:scstatususerid>admin</ccp:scstatususerid>
<ccp:sourcetype/>
<ccp:sctags>
  <ccp:sctag>tag14</ccp:sctag>
</ccp:sctags>
</entry>
</feed>

```

## Social Contact API Parameters

Parameters are optional unless otherwise noted.

| Parameter              | Description   | Notes  |
|------------------------|---|--|
| <b>author</b>          | The social contact author name.   | String.<br>Required for POST.  |
| <b>description</b>     | The body of the social contact.   | String.  |
| <b>extensionField</b>  | A wrapper tag for a custom name and value pair.   |  |
| <b>extensionFields</b> | A collection of custom name and value pairs.  | The person submitting the social contact may specify up to 100 pairs, and the entire collection can contain up to one megabyte of information.<br><br>To update user data, provide a new value for the existing extension field name. If you include an extension field element with the name but no value, the corresponding name and value pair will be deleted during the update. |
| <b>feedRefURL</b>      | The feed refURL that the social contact is associated with.                               | String.<br>Required for POST.  |
| <b>isSoftLocked</b>    | Indicates that the social contact should not be modified, but is not enforced in the API. | Boolean.<br><br>The SocialMiner UI will not permit any modifications via the UI to a social contact when isSoftLocked is set to true. This is normally used by SocialMiner when a social contact is to be or has been queued to an outside entity.   |



| Parameter                     | Description   | Notes  |
|-------------------------------|---|--|
| <b>isInvited</b>              | Indicates if the social contact was created as a result of a chat invitation.                   | Boolean.<br>Default is false.<br>If the Social Contact was created as a result of a chat invitation, then isInvited must be set to true.   |
| <b>inviteStatus</b>           | The status of chat invitations sent (if any) from this social contact.                          | String, case-insensitive.<br>Default is NONE.<br>Valid values are: <ul style="list-style-type: none"> <li>• NONE (no chat invitations were sent from this social contact)</li> <li>• SENT (chat invitations were sent from this social contact)</li> <li>• EXPIRED (chat invitations were sent but were not accepted by the customer)</li> </ul> |
| <b>publishedDate</b>          | The social contact published date.  | String.<br>Leave blank to use the current timestamp or provide a valid Unix timestamp.   |
| <b>refURL</b>                 | A copy of the URL requested.  |  |
| <b>replyTemplateRefURL</b>    | The reference URL of the reply template. This can be used to retrieve further template details. | Returned by update if the social contact is associated with a feed that has been configured to use a reply template.   |
| <b>replyTemplateURL</b>       | The URL of the reply template.  | Returned by update if the social contact is associated with a feed that has been configured to use a reply template.   |
| <b>shortUrlId/shortUrlIds</b> | Is a list of short URL Ids which were generated for this social contact.                        | String.<br>A short URL is generated by the system when a SocialMiner user sends a chat invitation to a customer. The social contact from which the invitation was sent maintains a short URL for each invitation.  |

| Parameter           | Description   | Notes  |
|---------------------|---|--|
| <b>sourceType</b>   | Is the type of feed this social contact came from: rss, facebook, twitter_stream, twitter_account, ,twitter_search, callback, chat, or push.  | String.<br><br>This is set by SocialMiner and cannot be set through the create or update APIs.   |
| <b>status</b>       | One of: <ul style="list-style-type: none"> <li>• <b>unread</b>—The default state of a new contact.</li> <li>• <b>reserved</b>—Reserved to be handled.</li> <li>• <b>handled</b>—This contact has been handled and no further action is required.</li> <li>• <b>discarded</b>—This contact does not require a response and is filed in the recycle bin.</li> <li>• <b>queued</b>—The contact is in the process of or has been routed to some external entity.</li> <li>• <b>draft</b>—A draft response to the contact has been created and saved, but not sent.</li> </ul> | String (case-sensitive).<br><br>If the submitted and the current status of the social contact are not equal, the submitted status becomes the effective status of the social contact.<br><br>The QUEUED status is settable from the API. However, currently, it is only set internally by SocialMiner. SocialMiner sets the social contact status to QUEUED when a social contact is to be or has been routed to some external entity. |
| <b>statusReason</b> | The reason why the contact is in the current state.   |  |
| <b>statusUserId</b> | The user modifying the status to a state other than UNREAD.   | The value changes to the user who is currently authenticated against the API.  |

| Parameter               | Description  | Notes   |
|-------------------------|--|---|
| <b>statusTimestamp</b>  | The time stamp of the last state change of the social contact. | Long integer.<br>Required for PUT.<br><b>Important:</b> You must provide the current statusTimestamp of the social contact when you perform an update. If you do not provide the same statusTimestamp as returned from a social contact <b>get</b> request, then the update fails. This mechanism is in place so that two clients cannot update the same social contact at the same time.<br>The statusTimestamp changes to the current <b>timestamp</b> if the update is successful. |
| <b>tag/tags</b>         | One or more tags to associate with this social contact.        | The tags can be new or existing tags. If you include the <i>tags</i> element, but do not include any <i>tag</i> elements, then tags are deleted during an update.   |
| <b>title</b>            | The title of the social contact.                               | String.<br>Required for POST.   |
| <b>transcriptRefURL</b> | Is a URL to get the chat transcript.                           | String.<br>Only applies to contacts where sourceType is chat.   |





# Tag

SocialMiner supports the labeling of contacts with tags. Tags can be added, edited, and removed to or from a social contact using the social contact API ([Social Contact](#), on page 155).

- [Tag API Command](#), page 169

## Tag API Command

This section describes the supported command (GET) for the tag API and the parameters for that command.

### GET (List)

List all configured tags that exist.

|                       |   |
|-----------------------|---|
| URL:                  | http://<ServerIP>:<Port>/ccp-webapp/ccp/tag   |
| HTTP method:          | GET   |
| Example XML response: | <pre>&lt;Tags&gt;   &lt;Tag&gt;     &lt;name&gt;tagname1&lt;/name&gt;   &lt;/Tag&gt;   &lt;Tag&gt;     &lt;name&gt;tagname2&lt;/name&gt;   &lt;/Tag&gt;   ... &lt;/Tags&gt;</pre> |





## Twitter Reply

The Twitter reply API allows you to respond to tweets or Twitter direct messages. You must configure a Twitter account, stream, or search feed before you can use this API.



### Note

The Shindig OpenSocial container in which SocialMiner runs requires that REST requests complete within five seconds. Communication with Twitter servers can exceed five seconds. This limitation means you must poll after making calls to the Twitter reply API to verify the status returned. A diagram is provided to illustrate the API calls and expected poll responses.

The [SocialMiner Troubleshooting Tips](#) has a list of common Twitter errors and their causes. Refer to [http://dev.twitter.com/pages/responses\\_errors](http://dev.twitter.com/pages/responses_errors) for additional information about Twitter errors.

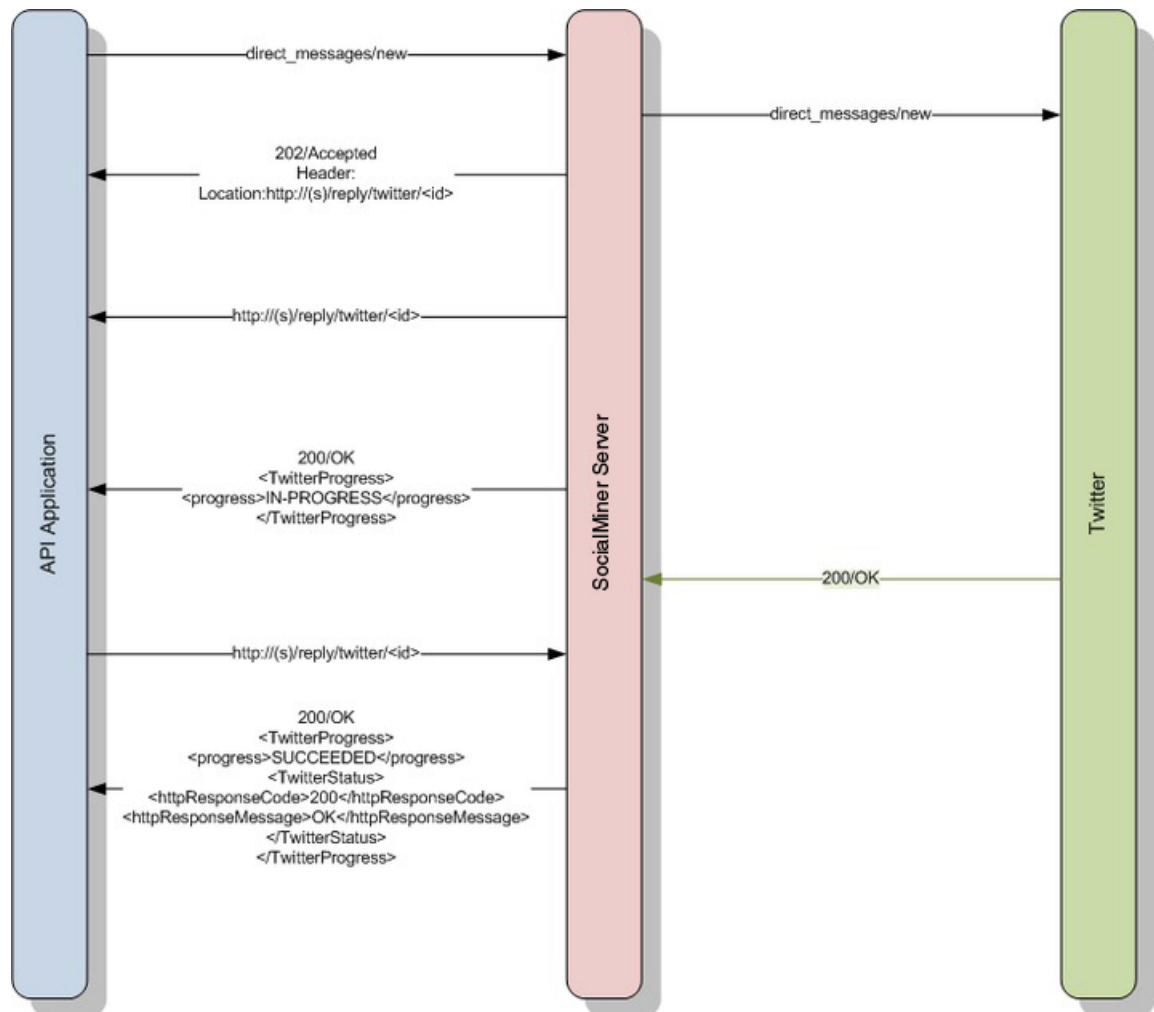
See also [Authorize Against Twitter Feeds](#), on page 75.

- [Twitter Follow API Flow](#), page 172
- [Twitter Reply API Commands](#), page 172

## Twitter Follow API Flow

This diagram illustrates the flow of API calls and expected poll responses for Twitter follow.

**Figure 3: Twitter Follow API Flow Diagram**



SocialMiner will wait for a response from Twitter for 30 seconds. If SocialMiner does not receive a response within 30 seconds, SocialMiner will fail the request and return an http response code of 408.

## Twitter Reply API Commands

This section describes the supported commands for the Twitter reply API and the parameters for those commands.



**Related Topics**

[GET, on page 173](#)  
[GET \(User\), on page 174](#)  
[GET \(Friendships/Exists\), on page 175](#)  
[POST \(Create Status - Tweet\), on page 175](#)  
[POST \(Create Direct Message\), on page 176](#)  
[POST \(Create Retweet\), on page 177](#)  
[POST \(Create Follow\), on page 178](#)  
[POST \(Create Unfollow\), on page 178](#)

**GET**

Gets the status of a Twitter Reply API call.

|                          |  |
|--------------------------|--|
| <b>URL:</b>              | <p>http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/reply/twitter/&lt;id&gt;</p> <p>For more information about the elements in the URL, see <a href="#">API Conventions, on page 1</a>. In this instance, &lt;id&gt; represents the ProgressID being requested.</p>   |
| <b>HTTP method:</b>      | GET  |
| <b>Example response:</b> | <p>If the fields in the initial request are valid, the Location field in the response contains the URL for the social contact associated with the post being queried.</p> <p>If the operation succeeds, the response returns the following XML.</p> <pre>&lt;TwitterProgress&gt;   &lt;progress&gt;SUCCEEDED FAILED IN-PROGRESS&lt;/progress&gt;   &lt;TwitterStatus&gt;&lt;/TwitterStatus&gt;   &lt;httpResponseCode&gt;responseCode&lt;/httpResponseCode&gt;   &lt;httpResponseMessage&gt;responseMessage&lt;/httpResponseMessage&gt; &lt;/TwitterProgress&gt;</pre> <p>If the operation fails, the httpResponseCode and httpResponseMessage fields contain the code and message returned by Twitter. The apiErrors field can contain additional detailed information about the error.</p> |
| <b>Response payload:</b> | <p>The response includes the following fields:</p> <ul style="list-style-type: none"> <li>• <b>progress:</b> one of the following:           <ul style="list-style-type: none"> <li>◦ <b>SUCCEEDED:</b> the operation succeeded.</li> <li>◦ <b>FAILED:</b> the Twitter operation failed. Use httpResponseCode and httpResponseMessage to determine why the operation failed.</li> <li>◦ <b>IN-PROGRESS:</b> is waiting for a response from Twitter.</li> </ul> </li> <li>• <b>TwitterStatus:</b> varies depending on the API called. See examples for the individual Twitter Reply API methods.</li> <li>• <b>httpResponseCode:</b> the response code received from Twitter.</li> </ul>  |

- **httpResponseMessage**: the response message received from Twitter.
- **ApiErrors**: a list of errors describing the failure.
- **ApiError**: individual error details.
- **ErrorMessage**: the error details returned by Twitter.
- **ErrorType**: SocialMiner's translation of the error returned by Twitter. SocialMiner uses this field for internationalization and localization purposes.

## GET (User)

Retrieves the profile information of a given Twitter user. It works similarly to the way [GET](#) does.

|                              |   |
|------------------------------|---|
| <b>URL:</b>                  | http://<ServerIP>:<Port>/ccp-webapp/ccp/reply/twitter/users/show  |
| <b>HTTP method:</b>          | GET   |
| <b>Parameters:</b>           | <ul style="list-style-type: none"> <li>• <b>screenName</b>: the user name of the Twitter account.</li> <li>• <b>account_user</b>: this parameter is optional. If specified, the account_user must be associated with a Twitter account feed configured on SocialMiner. If this is the case, the users/show call is made to Twitter using the account user's OAuth credentials.</li> </ul>   |
| <b>Example http request:</b> | http://<ServerIP>:<Port>/ccp-webapp/ccp/reply/twitter/users/show?screenName=ccpdoctest  |
| <b>Example XML response:</b> | <p>If the fields in the initial request are valid, then the response header's location field contains the URL for the social contact associated with the post being queried.</p> <p>If the operation succeeds, the polling URL</p> <p>(http://&lt;ServerIP&gt;:&lt;Port&gt;/ccp-webapp/ccp/reply/twitter/8)</p> <p>contains the following XML.</p> <pre>&lt;TwitterProgress&gt;   &lt;progress&gt;SUCCEEDED&lt;/progress&gt;   &lt;twitterUser&gt;     &lt;screenName&gt;twitter_user&lt;/screenName&gt;     &lt;id&gt;1234567890&lt;/id&gt;     &lt;name&gt;A. Twitter User&lt;/name&gt;     &lt;description&gt;&lt;/description&gt;     &lt;profileImageUrl&gt;http://a2.twimg.com/profile_images/60201051/WileECoyote_normal.jpg&lt;/profileImageUrl&gt;     &lt;url&gt;http://www.mycompany.com/&lt;/url&gt;     &lt;followersCount&gt;10&lt;/followersCount&gt;   &lt;/twitterUser&gt;   &lt;httpResponseCode&gt;responseCode&lt;/httpResponseCode&gt;   &lt;httpResponseMessage&gt;responseMessage&lt;/httpResponseMessage&gt; &lt;/TwitterProgress&gt;</pre> <p>If the operation fails, the httpResponseCode and httpResponseMessage fields will contain the code and message returned by Twitter.</p> |

## GET (Friendships/Exists)

Determines if user\_a is following user\_b.

|                              |   |
|------------------------------|---|
| <b>URL:</b>                  | http://<ServerIP>:<Port>/ccp-webapp/ccp/reply/twitter/friendships/exists  |
| <b>HTTP method:</b>          | GET   |
| <b>Parameters:</b>           | <ul style="list-style-type: none"> <li>• <b>user_a</b>: the screen name of the user who may or may not be following user_b</li> <li>• <b>user_b</b>: the screen name of the user to may or may not be followed by user_a</li> <li>• <b>account_user</b>: this parameter is optional. If specified, the account_user must be associated with a Twitter account feed configured on SocialMiner. If this is the case, the users/show call is made to Twitter using the account user's OAuth credentials.</li> </ul>  |
| <b>Example http request:</b> | http://<ServerIP>:<Port>/ccp-webapp/ccp/reply/twitter/users/show?screenName=ccpdoctest  |
| <b>Example XML response:</b> | <p>If the fields in the initial request are valid, then the response header's location field contains the URL for the social contact associated with the post being queried.</p> <p>If the operation succeeds, the polling URL</p> <p>(http://192.168.0.1/ccp-webapp/ccp/reply/twitter/8)</p> <p>contains the following XML.</p> <pre>&lt;TwitterProgress&gt;   &lt;progress&gt;SUCCEEDED&lt;/progress&gt;   &lt;TwitterStatus&gt;     &lt;friends&gt;true false&lt;/friends&gt;   &lt;/TwitterStatus&gt;   &lt;httpResponseCode&gt;responseCode&lt;/httpResponseCode&gt;   &lt;httpResponseMessage&gt;responseMessage&lt;/httpResponseMessage&gt; &lt;/TwitterProgress&gt;</pre> <p>If the operation fails, the httpResponseCode and httpResponseMessage fields will contain the code and message returned by Twitter.</p> |

## POST (Create Status - Tweet)

Sends a Twitter status message (tweet) from a configured twitter account.

|                     |  |
|---------------------|--|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/reply/twitter/statuses/update  |
| <b>HTTP method:</b> | POST   |
| <b>Parameters:</b>  | <ul style="list-style-type: none"> <li>• <b>username</b>: the username of a configured twitter account. The twitter account must be configured on the system.</li> </ul> |

## POST (Create Direct Message)

|                                     |   |
|-------------------------------------|---|
|                                     | <ul style="list-style-type: none"> <li>• <b>message</b>: the text of the message (which is limited to 140 characters).</li> <li>• <b>inReplyToStatusId</b>: optional. If this tweet is in reply to another tweet, use this field to specify the ID of the original tweet.</li> </ul>  |
| <b>Example XML request payload:</b> | <pre>&lt;Status&gt;   &lt;username&gt;twitterScreenName&lt;/username&gt;   &lt;message&gt;Tweet text&lt;/message&gt; &lt;/Status&gt;</pre>  |
| <b>Response:</b>                    | <p>If the fields in the initial request are valid, then the response header's location field contains the URL for the social contact associated with the post being queried.</p> <p>If the operation succeeds, the polling URL contains the following XML:</p> <pre>&lt;TwitterProgress&gt;   &lt;progress&gt;SUCCEEDED&lt;/progress&gt;   &lt;TwitterStatus&gt;     &lt;id&gt;idAssignedByTwitter&lt;/id&gt;     &lt;text&gt;textOfTheTweet&lt;/text&gt;     &lt;user&gt;       &lt;screenName&gt;usersScreenName&lt;/screenName&gt;       &lt;id&gt;userIdAssignedByTwitter&lt;/id&gt;     &lt;/user&gt;     &lt;createdAt&gt;tweetCreateDate&lt;/createdAt&gt;     &lt;inReplyToScreenName&gt;screenNameOfReplyRecipient&lt;/inReplyToScreenName&gt;      &lt;inReplyToStatusId&gt;idOfOriginalTweet&lt;/inReplyToStatusId&gt;     &lt;inReplyToUserId&gt;idOfReplyRecipient&lt;/inReplyToUserId&gt;   &lt;/TwitterStatus&gt;   &lt;httpResponseCode&gt;responseCode&lt;/httpResponseCode&gt;   &lt;httpResponseMessage&gt;responseMessage&lt;/httpResponseMessage&gt; &lt;/TwitterProgress&gt;</pre> <p>If the operation fails, the httpResponseCode and httpResponseMessage fields contain the code and message returned by Twitter.</p> |

## POST (Create Direct Message)

Sends a Twitter Direct Message (DM) from a configured twitter account.

|                     |  |
|---------------------|--|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/reply/twitter/direct_messages/new  |
| <b>HTTP method:</b> | POST   |
| <b>Parameters:</b>  | <ul style="list-style-type: none"> <li>• <b>fromUsername</b>: the configured Twitter account username for the user sending the message.</li> <li>• <b>toUsername</b>: the twitter username of the recipient of the DM.</li> <li>• <b>message</b>: the text of the message (which is limited to 140 characters).</li> </ul> |

|                                     |   |
|-------------------------------------|---|
| <b>Example XML request payload:</b> | <pre> &lt;DirectMessage&gt;   &lt;fromUsername&gt;TwitterAccountUsernam&lt;/fromUsername&gt;   &lt;toUsername&gt;someTwitterAccount&lt;/toUsername&gt;   &lt;message&gt;This is a test of the create direct message API.&lt;/message&gt; &lt;/DirectMessage&gt; </pre>  |
| <b>Response:</b>                    | <p>If the fields in the initial request are valid, then the response header's location field contains the URL for the social contact associated with the post being queried.</p> <p>If the operation succeeds, the polling URL contains the following XML.</p> <pre> &lt;TwitterProgress&gt;   &lt;progress&gt;SUCCEEDED&lt;/progress&gt;   &lt;TwitterStatus&gt;     &lt;id&gt;idAssignedByTwitter&lt;/id&gt;     &lt;text&gt;textOfTheTweet&lt;/text&gt;     &lt;createdAt&gt;tweetCreateDate&lt;/createdAt&gt;     &lt;senderScreenName&gt;senderScreenName&lt;/senderScreenName&gt;     &lt;recipientScreenName&gt;recipientScreenName&lt;/recipientScreenName&gt;   &lt;/TwitterStatus&gt;   &lt;httpResponseCode&gt;responseCode&lt;/httpResponseCode&gt;   &lt;httpResponseMessage&gt;responseMessage&lt;/httpResponseMessage&gt; &lt;/TwitterProgress&gt; </pre> <p>If the operation fails, the httpResponseCode and httpResponseMessage fields contain the code and message returned by Twitter.</p> |

## POST (Create Retweet)

Retweets a Twitter social contact from a configured Twitter account.

|                                     |  |
|-------------------------------------|--|
| <b>URL:</b>                         | http://<ServerIP>:<Port>/ccp-webapp/ccp/reply/twitter/statuses/retweet   |
| <b>HTTP method:</b>                 | POST   |
| <b>Parameters:</b>                  | <ul style="list-style-type: none"> <li>• <b>username:</b> the username of the configured Twitter account sending the retweet.</li> <li>• <b>tweetId:</b> the Twitter ID of the tweet.</li> </ul>   |
| <b>Example XML request payload:</b> | <pre> &lt;Status&gt;   &lt;username&gt;twitterScreenName&lt;/username&gt;   &lt;tweetId&gt;Tweet text&lt;/tweetId&gt; &lt;/Status&gt; </pre>   |
| <b>Response:</b>                    | <p>If the fields in the initial request are valid, then the response header's location field contains the URL for the social contact associated with the post being queried.</p> <p>If the operation succeeds, the polling URL contains the following XML:</p> <pre> &lt;TwitterProgress&gt;   &lt;progress&gt;SUCCEEDED&lt;/progress&gt;   &lt;TwitterStatus&gt;     &lt;id&gt;idAssignedByTwitter&lt;/id&gt;     &lt;text&gt;textOfTheTweet&lt;/text&gt;     &lt;user&gt;       &lt;screenName&gt;usersScreenName&lt;/screenName&gt;       &lt;id&gt;userIdAssignedByTwitter&lt;/id&gt;     &lt;/user&gt;   &lt;/TwitterStatus&gt; &lt;/TwitterProgress&gt; </pre> |

```

<createdAt>tweetCreateDate</createdAt>
<inReplyToScreenName>screenNameOfReplyRecipient</inReplyToScreenName>

<inReplyToStatusId>idOfOriginalTweet</inReplyToStatusId>
<inReplyToUserId>idOf ReplyRecipient</inReplyToUserId>
</TwitterStatus>
<httpResponseCode>responseCode</httpResponseCode>
<httpResponseMessage>responseMessage</httpResponseMessage>
</TwitterProgress>

```

If the operation fails, the `httpResponseCode` and `httpResponseMessage` fields contain the code and message returned by Twitter.

## POST (Create Follow)

Follows a user from a configured Twitter account feed.


|                     |   |
|---------------------|---|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/reply/twitter/follow  |
| <b>HTTP method:</b> | POST  |
| <b>Parameters:</b>  | <p>This method uses post but uses query parameters to specify who to follow and who will follow:</p> <ul style="list-style-type: none"> <li>• <b>account_user:</b> the twitter account feed user. This username must match one of the user names in a Twitter account feed.</li> <li>• <b>user_to_follow:</b> the screen name of the Twitter user to follow.</li> </ul>   |
| <b>Response:</b>    | <p>If the fields in the initial request are valid, then the response header's location field contains the URL for the social contact associated with the post being queried.</p> <p>If the operation succeeds, the polling URL contains the following XML:</p> <pre> &lt;TwitterProgress&gt;   &lt;progress&gt;SUCCEEDED&lt;/progress&gt;   &lt;httpResponseCode&gt;responseCode&lt;/httpResponseCode&gt;   &lt;httpResponseMessage&gt;responseMessage&lt;/httpResponseMessage&gt; &lt;/TwitterProgress&gt; </pre> <p>If the operation fails, the <code>httpResponseCode</code> and <code>httpResponseMessage</code> fields contain the code and message returned by Twitter.</p> |

## POST (Create Unfollow)

Stops following a user from a configured Twitter account feed.

|                     |  |
|---------------------|--|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/reply/twitter/unfollow                                       |
| <b>HTTP method:</b> | POST   |
| <b>Parameters:</b>  | <p>This method uses post but uses query parameters to specify who to follow and who will follow.</p> |

|                  |   |
|------------------|---|
|                  | <ul style="list-style-type: none"><li>• <b>account_user</b>: the twitter account feed user. This username must match one of the user names in a Twitter account feed.</li><li>• <b>user_to_unfollow</b>: the screen name of the Twitter user to stop following.</li></ul>   |
| <b>Response:</b> | <p>If the fields in the initial request are valid, then the response header's location field contains the URL for the social contact associated with the post being queried.</p> <p>If the operation succeeds, the polling URL contains the following XML:</p> <pre>&lt;TwitterProgress&gt;   &lt;progress&gt;SUCCEEDED&lt;/progress&gt;   &lt;httpResponseCode&gt;responseCode&lt;/httpResponseCode&gt;   &lt;httpResponseMessage&gt;responseMessage&lt;/httpResponseMessage&gt; &lt;/TwitterProgress&gt;</pre> <p>If the operation fails, the httpResponseCode and httpResponseMessage fields contain the code and message returned by Twitter.</p> |

 POST (Create Unfollow)





# URL Shortener

The URL shortener API provides a shortened version of a longer URL to the public. The shortened URL has an expiration time and may only be used once. Shortened URLs may not be modified after they are created.

There are two supported types for ShortURL: generic and chat\_invite. The default type is generic when a short URL is created with no <type> field explicitly provided in the XML body.



**Note**

All time stamps should be expressed as milliseconds since January 1,1970 in UTC/GMT.

See also [Public URL Prefix for Chat Invitation](#) for information on building the full URL.

- [URL Shortener API Commands](#), page 181

## URL Shortener API Commands

This section describes the supported commands for the URL shortener API and the parameters for those commands.

### POST

Creates a shortened URL.

Based on the type of shortURL created, there are different required fields: generic shortURLs require only url; chat\_invite shortURLs require url, campaignRefURL and scRefURL.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/shorturl/   |
| <b>HTTP method:</b> | POST  |
| <b>Parameters:</b>  | <p><b>Type:</b> optional. The shortURL type. Valid values are generic (the default) and chat_invite.</p> <p><b>URL:</b> required. The URL being shortened (required).</p> |

|                                     |  |
|-------------------------------------|--|
|                                     | <p><b>campaignRefURL:</b> required for chat_invite type. The refURL of the campaign for which this short URL is being created.</p> <p><b>scRefURL:</b> required for chat_invite type. The refURL of the inviting social contact.</p> <p><b>active:</b> optional. For the chat_invite type, active means that the shortURL was successfully sent with the reply as a chat invitation. Valid values are false (the default) or true.</p> <p><b>expireDate:</b> optional. The expiration date of the shortened URL. Defaults to 24 hours from creation. Expiration may be no more than 30 days in the future.</p>   |
| <b>Example XML request payload:</b> | <pre>&lt;ShortURL&gt;   &lt;type&gt;chat_invite&lt;/type&gt;   &lt;url&gt;http://theurl.com?paraml=val1&lt;/url&gt;   &lt;campaignRefURL&gt;http://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/     [public_ID]&lt;/campaignRefURL&gt;   &lt;scRefURL&gt;http://[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/     [SC_ID]&lt;/scRefURL&gt;   &lt;expireDate&gt;[timestamp]&lt;/expireDate&gt; &lt;/ShortURL&gt;</pre> <p>Or, for a generic shortURL:</p> <pre>&lt;ShortURL&gt;   &lt;url&gt;http://theurl.com?paraml=val1&lt;/url&gt; &lt;/ShortURL&gt;</pre> <p><b>Note</b> When using a full url with multiple parameters, special characters must be properly escaped. This means, for example, that characters such as "&amp;" should appear inside the xml body as "&amp;amp;".</p> |
| <b>HTTP response headers:</b>       | <p>If successful, the location field in the http response header will have a URL to the newly created short URL. A GET of the newly created short URL will provide the shortened URL.</p> <p>See <a href="#">API Conventions</a> for error information.</p>  |

## GET

Get a shortened URL.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | http:// <ServerIP>:<Port> /ccp-webapp/ccp/shorturl/<id> |
| <b>HTTP method:</b> | GET   |

|                                      |   |
|--------------------------------------|---|
| <b>Example XML response payload:</b> | <pre> &lt;ShortURL&gt;   &lt;type&gt;chat_invite&lt;/type&gt;   &lt;active&gt;false&lt;/active&gt;   &lt;url&gt;http://theurl.com?param1=val1&lt;/url&gt;   &lt;creator&gt;[userID]&lt;/creator&gt;   &lt;campaignRefURL&gt;http://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/[public ID]&lt;/campaignRefURL&gt;   &lt;scRefURL&gt;http://[ServerIP]:[Port]/ccp/socialcontact/[SC ID]&lt;/scRefURL&gt;   &lt;createdDate&gt;[timestamp]&lt;/createdDate&gt;   &lt;expireDate&gt;[timestamp]&lt;/expireDate&gt;   &lt;usedDate&gt;[timestamp]&lt;/usedDate&gt;   &lt;shortURL&gt;/ccp/s/[id]&lt;/shortURL&gt;   &lt;refURL&gt;http://[ServerIP]:[Port]/ccp-webapp/ccp/shorturl/[id]&lt;/refURL&gt; &lt;/ShortURL&gt; </pre> <p>The "refURL" is a copy of the URL requested.</p>   |
| <b>Elements:</b>                     | <p><b>type:</b> the shortURL type.</p> <p><b>URL:</b> the URL being shortened.</p> <p><b>creator:</b> user that created the short URL.</p> <p><b>active:</b> for the chat_invite type, active means that the shortURL was successfully sent with the reply as a chat invitation.</p> <p><b>campaignRefURL:</b> the refURL of the campaign for which this short URL was created.</p> <p><b>scRefURL:</b> the refURL of the inviting social contact.</p> <p><b>createdDate:</b> timestamp on which short URL was created.</p> <p><b>expireDate:</b> the expiration date of the shortened URL.</p> <p><b>usedDate:</b> the timestamp on which this URL was used. Empty if not used yet.</p> <p><b>shortURL:</b> the shortened URL absolute path on the SocialMiner server.</p> <p><b>refURL:</b> the URL of this short URL object.</p> |
| <b>HTTP response headers:</b>        | See <a href="#">HTTP Responses</a> .  |

## GET (List)

List all shortened URLs.

|                     |   |
|---------------------|---|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/shorturl/ |
| <b>HTTP method:</b> | GET   |

**Example XML request payload:**

```

<shortURLs>

  <ShortURL>
    <type>generic</type>
    <active>false</active>
    <url>http://theurl.com?param1=val1</url>
    <creator>[userID]</creator>
    <createdDate>[timestamp]</createdDate>
    <expireDate>[timestamp]</expireDate>
    <usedDate>[timestamp]</usedDate>
    <shortURL>/ccp-webapp/ccp/s/[id]</shortURL>
    <refURL>http://[ServerIP]:[Port]
      /ccp-webapp/ccp/shorturl/[id]</refURL>
  </ShortURL>

  <ShortURL>
    <type>chat_invite</type>
    <active>true</active>
    <url>http://cisco.com/index.htm</url>
    <creator>[userID]</creator>
    <campaignRefURL>http://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/
      [public ID]</campaignRefURL>
    <scRefURL>http://[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/
      [SC ID]</scRefURL>
    <expireDate>[timestamp]</expireDate>
    <usedDate>[timestamp]</usedDate>
    <shortURL>/ccp-webapp/ccp/s/[id]</shortURL>
    <refURL>http://[ServerIP]:[Port]
      /ccp-webapp/ccp/shorturl/[id]</refURL>
  </ShortURL>

</shortURLs>

```

## PUT (update)

Update a shortened URL.

Only two fields can be updated on an already created shortened URL : active and usedDate.

Any attempt to update other fields will result in an error.

|                                     |  |
|-------------------------------------|--|
| <b>URL:</b>                         | http://<ServerIP>:<Port>/ccp-webapp/ccp/shorturl/<id>  |
| <b>HTTP method:</b>                 | PUT  |
| <b>Example XML request payload:</b> | <pre> &lt;ShortURL&gt;   &lt;active&gt;true&lt;/active&gt;   &lt;usedDate&gt;[timestamp]&lt;/usedDate&gt; &lt;/ShortURL&gt; </pre> |
| <b>HTTP response headers:</b>       | See <a href="#">HTTP Responses</a> .   |

## DELETE

Delete a shortened URL.

|             |   |
|-------------|---|
| <b>URL:</b> | http://<ServerIP>:<Port>/ccp-webapp/ccp/shorturl/<id> |
|-------------|---|

|                               |  |
|-------------------------------|--|
| <b>HTTP method:</b>           | DELETE   |
| <b>HTTP response headers:</b> | See <a href="#">API Conventions</a> for error information. |

 DELETE



## XMPP

The XMPP API allows an administrator to retrieve the existing XMPP server configuration and to update it if necessary. An XMPP server connection is required to send Instant Messaging (IM) notifications.

This API is represented on the SocialMiner user interface in the System Administration panel. Only one XMPP configuration of both Server and User is allowed at this time.



### Note

Only the administrator created during install can use this API.

- [XMPP API Commands, page 187](#)

## XMPP API Commands

This section describes the supported commands for the XMPP API and the parameters for those commands.

### Related Topics

[GET, on page 187](#)

[PUT, on page 188](#)

[XMPP API Parameters, on page 188](#)

## GET

Get the XMPP configuration.

|                     |  |
|---------------------|--|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/xmpp/default |
| <b>HTTP method:</b> | GET  |

|                            |  |
|----------------------------|--|
| <b>Example XML output:</b> | <pre> &lt;Xmpp&gt;   &lt;xmppService&gt;xmpp.cisco.com&lt;/xmppService&gt;   &lt;xmppHost&gt;[ServerIP]&lt;/xmppHost&gt;   &lt;xmppPort&gt;[Port]&lt;/xmppPort&gt;   &lt;xmppServiceLookup&gt;true&lt;/xmppServiceLookup&gt;   &lt;xmppEnabled&gt;true&lt;/xmppEnabled&gt;   &lt;xmppServiceUsername&gt;gding&lt;/xmppServiceUsername&gt;   &lt;xmppServicePassword&gt;...&lt;/xmppServicePassword&gt;   &lt;refURL&gt;     http://[ServerIP]:[Port]/ccp-webapp/ccp/     xmpp/default   &lt;/refURL&gt; &lt;/Xmpp&gt; </pre> |
| <b>Parameters:</b>         | See <a href="#">XMPP API Parameters, on page 188</a> .   |

## PUT

Updates the XMPP configuration.

|                     |  |
|---------------------|--|
| <b>URL:</b>         | http://<ServerIP>:<Port>/ccp-webapp/ccp/xmpp/default   |
| <b>HTTP method:</b> | PUT  |
| <b>Parameters:</b>  | See <a href="#">XMPP API Parameters, on page 188</a> . |

## XMPP API Parameters

All parameters are optional.

| Parameter          | Description  | Notes  |
|--------------------|--|--|
| <b>xmppEnabled</b> | Identifies whether this XMPP configuration is enabled or disabled. | Boolean. Default is false.   |
| <b>xmppHost</b>    | The IP address or hostname of the XMPP server.                     | String.  |
| <b>xmppPort</b>    | The XMPP port number. The default port is 5222.                    | Integer.<br>Not used if <i>xmppServiceLookup</i> is set to <i>true</i> . |
| <b>xmppService</b> | The xmpp service lookup name.                                      | String.  |



| Parameter                  | Description   | Notes                     |
|----------------------------|---|---------------------------|
| <b>xmppServiceLookup</b>   | If this flag is true, the xmppService field will be used to perform a DNS Service lookup for the XMPP Service; otherwise the xmppHost and xmppPort will be used to directly connect to the XMPP server. | Boolean. Default is true. |
| <b>xmppServiceUserName</b> | The username used to log into the XMPP server.  | String.                   |
| <b>xmppServicePassword</b> | The password used to log into the XMPP server.  | String.                   |





## APPENDIX

# A

## Reporting Database Connection

---

You connect to the reporting database using Java database connectivity (JDBC). The reporting database runs on Informix.

- [Reporting Database SQL Connection, page 191](#)
- [Reporting Database Schema, page 192](#)

## Reporting Database SQL Connection

Connection to the SocialMiner Informix reporting database is made using the following format:

```
jdbc:informix-sqli://<hostname>:<port>/  
<databaseName>:INFORMIXSERVER=<informixserver>;
```

Where:

- the reporting database <port> is 1526.
- the <databaseName> is "mmca\_data".
- the <informixserver> name is based on the hostname of the server with `_mmca` append to the end of the hostname. Also, any dashes ("-") in the hostname are replace by underscores ("\_").

For example, if your server hostname is *my-server.com*, then the INFORMIXSERVER name is *my\_server\_mmca*. The complete JDBC URL would be:

```
jdbc:informix-sqli://my-server.com:1526/  
mmca_data:INFORMIXSERVER=my_server_mmca;
```



### Note

---

When authenticating, the username is always *reportinguser* and the password is the password you created in the Administration panel.

---

# Reporting Database Schema

The reporting database schema consists of the following tables:

- **mmca\_report\_campaign**
- **mmca\_campaign\_activity**
- **mmca\_agent\_campaign\_activity**

The mmca\_report\_campaign table contains information used in reports. It is synchronized with campaigns in the Configuration Database when campaign synchronization jobs are run.

**Table 1: mmca\_report\_campaign**

| Field name:             | Description:   | Data type:                   | Constraints:           |
|-------------------------|--|------------------------------|------------------------|
| <b>campaignid</b>       | Auto-incrementing surrogate ID                             | serial (8)                   | Primary key, not Null. |
| <b>configcampaignid</b> | The internal database ID.                                  | int (8)                      | Not Null.              |
| <b>campaignname</b>     | The campaign name, as defined in the campaign panel.       | nvarchar                     | Not Null.              |
| <b>lastupdated</b>      | Last time this row was updated.                            | datetime                     | Not Null.              |
| <b>active</b>           | Indicates if the campaign exists in the campaign database. | int; 1 = exists, 0 = deleted | Not Null.              |

The mmca\_campaign\_activity table is an aggregate table used for reporting campaign statistics.

**Table 2: mmca\_campaign\_activity**

| Field name:     | Description:  | Data type: | Constraints:           |
|-----------------|---|------------|------------------------|
| <b>recordid</b> | Auto-incrementing ID  | serial(8)  | Primary key, not Null. |
| <b>interval</b> | Reporting interval to which this record applies (currently only a 15 minute interval is supported). | datetime   | Not Null.              |

| Field name:         | Description:  | Data type: | Constraints: |
|---------------------|---|------------|--------------|
| <b>campaignid</b>   | ID of the campaign from the mmca_reportcampaign table.  | int(8)     | Not Null.    |
| <b>screceived</b>   | The number of social contacts that were received for this campaign for this interval.   | int(8)     | Not Null.    |
| <b>screerved</b>    | The number of social contacts that were reserved for this campaign for this interval.   | int        | Not Null.    |
| <b>schanded</b>     | The number of social contacts that were handled for this campaign for this interval.  | int        | Not Null.    |
| <b>scdiscarded</b>  | The number of social contacts that were discarded for this campaign for this interval.  | int        | Not Null.    |
| <b>reservedtime</b> | Cumulative reserved time for all social contacts reserved in this campaign for this interval.<br><br>Reserved time is the time between when the contact was received and when the contact was marked as reserved. | bigint     | Not Null.    |
| <b>handledtime</b>  | Cumulative handled time for all social contacts handled in this campaign for this interval.<br><br>Handled time is the time between when the contact was received and when the contact was marked as handled.     | bigint     | Not Null     |

| Field name:                   | Description:  | Data type: | Constraints: |
|-------------------------------|---|------------|--------------|
| <b>discardedtime</b>          | Cumulative discard time for all social contacts discarded in this campaign for this interval.<br><br>Discard time is the time between when the contact was received and when the contact was marked as discarded. | bigint     | Not Null     |
| <b>chatinvitationssent</b>    | The number of chat invitations sent from this campaign during the interval (whether they end up being handled in this campaign or not).   | int        | Not Null     |
| <b>chatinvitationshandled</b> | The number of chat invitations handled within this campaign during the interval (whether sent from this campaign or not).   | int        | Not Null     |
| <b>chatinvitationsexpired</b> | The number of chat invitations sent from this campaign that expired during the interval (the customer didn't click the chat invitation link before the invitation timed out).                                     | int        | Not Null     |

The `mmca_agent_campaign_activity` table is an aggregate table used for reporting agent-related campaign statistics.

**Table 3: `mmca_agent_campaign_activity`**

| Field name:     | Description:  | Data type: | Constraints:           |
|-----------------|---|------------|------------------------|
| <b>recordid</b> | Auto-incrementing ID  | serial(8)  | Primary key, not Null. |
| <b>interval</b> | Reporting interval to which this record applies (currently only a 15 minute interval is supported). | datetime   | Not Null.              |

| Field name:                | Description:   | Data type: | Constraints: |
|----------------------------|--|------------|--------------|
| <b>campaignid</b>          | ID of the campaign from the mmca_reportcampaign table.   | int(8)     | Not Null.    |
| <b>userid</b>              | String representing the login name of the user who modified this record for this interval.   | varchar    | Not Null.    |
| <b>schandled</b>           | Number of social contacts handled in this interval by the userid for this campaign during this interval.   | int        | Not Null.    |
| <b>scdiscarded</b>         | Number of social contacts discarded in this interval by the userid for this campaign.  | int        | Not Null.    |
| <b>screserveddiscarded</b> | Number of discarded social contacts in this interval that were previously reserved (at any time) by this user.   | int        | Not Null.    |
| <b>screservedhandled</b>   | Number of handled social contacts in this interval that were previously reserved (at any time) by this user.   | int        | Not Null.    |
| <b>handledtime</b>         | Cumulative handled time for all social contacts handled by this user in this interval for this campaign.<br><br>Handled time is defined as the time between when a contact was marked as reserved and the time the contact was marked handled. | int(8)     | Not Null.    |

| Field name:                   | Description:  | Data type: | Constraints: |
|-------------------------------|---|------------|--------------|
| <b>discardedtime</b>          | <p>Cumulative discarded time for all social contacts discarded by this user in this interval for this campaign.</p> <p>Discard time is defined as the time between when a contact was marked as reserved and the time the contact was marked discarded.</p> | int(8)     | Not Null.    |
| <b>chatinvitationssent</b>    | The number of chat invitations sent by the user from this campaign during the interval (whether they end up being handled in this campaign or not).   | int        | Not Null.    |
| <b>chatinvitationshandled</b> | The number of chat invitations handled by the user within this campaign during the interval (whether sent from this campaign or not).   | int        | Not Null.    |
| <b>chatinvitationsexpired</b> | The number of chat invitations sent by the user from this campaign that expired during the interval (the customer didn't click the chat invitation link before the invitation timed out).   | int        | Not Null.    |





## SocialMiner Server Configuration

- [Security Configuration Options](#) , page 197

### Security Configuration Options

SocialMiner may be deployed where some users access the server through a firewall or proxy, and others do not. For the customer chat interface, it is possible to prevent the SocialMiner server from being abused or limiting access for those outside the firewall to specific server functionality by deploying it behind a firewall or proxy server. Within one deployment, a reverse proxy or a firewall may be used — but not both.

### Port-Forward Firewall Configuration

When placed behind a port-forwarding firewall, the SocialMiner server is only reachable (for some users) by going through a specific port on a specific machine. All traffic on that port is forwarded to SocialMiner, and there is no alteration of the user's request as it traverses the firewall.

Any port may be chosen through which to forward traffic. Typically this would be port 80 or 443 (for http and https respectively), but there are no restrictions. For a port intended to forward non-SSL (http) traffic, the destination should be port 80 on the SocialMiner server. SSL (https) traffic should be forwarded to port 443.

There is no additional configuration required on the SocialMiner server.

### Reverse Proxy

A reverse proxy is used to forward specific requests to SocialMiner. During proxying, request headers are altered so that the proxied server has enough original request information to correctly create the served content (for example, so that links reference the proxy host and not the SocialMiner server). http or https may be used at the proxy server and requests may be forwarded to SocialMiner using either http or https.

The customer chat interface and URL redirect interfaces are supported for reverse proxying.

SocialMiner recognizes the following reverse proxy headers:

| Header                   | Required? | Comments   |
|--------------------------|-----------|--|
| <b>X-Forwarded-Host</b>  | Y         | Includes the proxy host name as visible to the user. May also include a port in the form <server name>:<port>  |
| <b>X-Forwarded-Proto</b> | N         | If present, determines the protocol of generated links. Defaults to http unless the proxy port is determined to be 443, in which case it will be https. Overrides Front-End-https and X-Forwarded-https values when present. |
| <b>X-Forwarded-Port</b>  | N         | If present, is returned by subsequent calls to Request.getServerPort(). If this header is present and a port is provided in X-Forwarded-Host, this value is overridden by the X-Forwarded-Host value.                        |
| <b>Front-End-https</b>   | N         | If present and value is "on", returned links will use https. It overrides X-Forwarded-https when present.  |
| <b>X-Forwarded-https</b> | N         | If present and value is "on", returned links will use https.   |

**Note**

For Apache users, by default Apache will not indicate when SSL was used to reach the proxy server. In order for SocialMiner links to be correctly formatted when SSL is being used between the user browser and the Apache reverse proxy, you must add a request header to proxied requests to tell SocialMiner to use https. You can do this by adding the following to your server configuration:

```
RequestHeader set X-Forwarded-Proto "https"
```



## XMPP BOSH Eventing

SocialMiner sends asynchronous state change and tag update events to an XMPP client using the XMPP Publish-Subscribe protocol ([XEP-0060](#)).

**Authentication:** Only SocialMiner authorized users are allowed to connect to the embedded XMPP server.

**Ports:** Connect on these ports for eventing:

- Port 7071 for unsecure XMPP BOSH connections
- Port 7443 for secure XMPP BOSH connections

**Domain:** SocialMiner uses a domain of 127.0.0.1

**JID:** The JID resulting from a connection to the SocialMiner XMPP server is formatted as follows:

<socialminerusername>@127.0.0.1/<resourceId>

**To address:** Set the to address used in subscriptions to pubsub.127.0.0.1

**Sample Subscription:**

```
<body xmlns="http://jabber.org/protocol/httpbind" sid="bf75bac1" rid="760657976">
  <iq id="iq13953502014548" xmlns="jabber:client" type="set" to="pubsub.127.0.0.1">
    <pubsub xmlns="http://jabber.org/protocol/pubsub">
      <subscribe node="ccp.campaign.updates.campaign1"
jid="administrator@127.0.0.1/bf75bac1"></subscribe>
    </pubsub>
  </iq>
</body>
```

- [Publish and Subscribe](#), page 199

## Publish and Subscribe

SocialMiner's event mechanism uses XMPP extensions for event subscription and publication. Details can be found here: <http://xmpp.org/extensions/xep-0060.html>.

When a tag is created or modified or when a contact state changes, the information is published using XMPP. The campaign results panel subscribes to the specific XMPP topic for the selected campaign, receives the change events, and updates the user interface appropriately.

## Nodes

In XMPP, publishers publish events to a node. Subscribers subscribe to nodes in order to receive events related to the node. Nodes are string-carried in the XML used to publish and subscribe. These strings are also carried in the notifications sent to subscribers.

Creating a campaign creates a node to allow subscribers to subscribe to events related to the campaign results. This node has the form *ccp.campaign.updates.<campaignpublicId>*, where *campaignpublicId* is the *publicId* field returned by a campaign's GET request.

When a campaign is deleted, the corresponding node is also deleted.

SocialMiner also creates a global node "ccp.contacts.chat" to publish event related to chat contacts.

## Events

### Global Chat Contact Events

| Field           | Description   |
|-----------------|---|
| id              | The unique ID of the contact.   |
| author          | The author of the contact's status (from the author field of the contact).                  |
| title           | The title of the contact's status (from the title field of the contact).                    |
| status          | The status of the contact.  |
| statusUserId    | The user who most recently changed the contact status.                                      |
| statusReason    | The reason the contact is in the current state.   |
| statusTimeStamp | The time at which the contact's status was changed.   |
| publishDate     | The date when SocialMiner received the contact.   |
| tags            | The list of tags associated with the contact.   |
| refURL          | The REST reference URL of the contact.  |
| chatIsInvited   | A boolean value to indicate whether this chat was initiated using a chat invitation or not. |

When an application subscribes to *ccp.contacts.chat node*, it receives events when a contact is updated in one of the following ways:

- The contact's tags are modified. (This happens any time the contact's update REST API includes the tags field.)
- The contact's status is modified.

The actual payload of the XML event is as follows:

```
<SocialContact xmlns="http://jabber.org/protocol/pubsub">
  <author>author1</author>
  <title>title1</title>
  <id>DA476CF81000012F000002FB0A568DF5</id>
  <publishedDate>1305037194000</publishedDate>
  <refURL>http://[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/
    DA476CF81000012F000002FB0A568DF5
  </refURL>
  <status>reserved</status>
  <statusTimestamp>1305037210727</statusTimestamp>
  <statusUserId>admin</statusUserId>
  <chatIsInvited>false</chatIsInvited>
  <tags>
    <tag>tag1</tag>
    <tag>tag2</tag>
  </tags>
</SocialContact>
```

### Campaign Contact Events

Each campaign results event contains the following attributes of a social contact:

| Field            | Description  |
|------------------|--|
| id               | The unique ID of the social contact.                       |
| status           | The status of the contact.                                 |
| statusUserId     | The user who most recently changed the contact status.     |
| statusTimeStamp  | The time at which the social contact's status was changed. |
| publishDate      | The date when SocialMiner received the contact.            |
| tags             | The list of tags associated with the contact.              |
| refURL           | The REST reference URL of the social contact.              |
| campaignpublicId | The ID of the campaign to which this contact belongs.      |

When an application subscribes to *campaign.updates*, it receives events when a social contact associated with that campaign changes in one of the following ways:

- The social contact's tags are modified. (This happens any time the social contact's update REST API includes the tags field.)
- The social contact's status is modified.

The actual payload of the XML event is as follows:

```
<SocialContact xmlns="http://jabber.org/protocol/pubsub">
  <campaignpublicId>
    EventingCampaign-07192-0000000000012
  </campaignpublicId>
  <id>DA476CF81000012F000002FB0A568DF5</id>
  <publishedDate>1305037194000</publishedDate>
```

```
<refURL>http://[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/  
DA476CF81000012F000002FB0A568DF5  
</refURL>  
<status>reserved</status>  
<statusTimestamp>1305037210727</statusTimestamp>  
<statusUserId>admin</statusUserId>  
<tags>  
  <tag>tag1</tag>  
  <tag>tag2</tag>  
</tags>  
</SocialContact>
```



## APPENDIX

# D

## Custom reply Templates

---

This information is intended for experienced web developers wishing to create a custom reply template. Developers should already be familiar with HTML and Javascript (including AJAX).

Custom reply templates give developers a way to provide extended functionality to users. Templates can interact with content on a variety of servers and can deliver rich user experiences; even embedding entire sub-applications within the SocialMiner interface.

SocialMiner provides a Javascript API to interact with the template container to access SocialMiner REST APIs, display messages, and close the template when finished. This section describes the primary files and functions used when developing custom reply templates. It also covers how to migrate templates written against pre-9.0.1 SocialMiner releases.

- [Javascript Concepts, page 203](#)

## Javascript Concepts

### Gadgets API

Custom reply templates are instances of OpenSocial gadgets (see <http://opensocial.org>). Gadgets are web pages that adhere to a certain format and that have access to a gadgets Javascript API. SocialMiner gadgets (including custom reply gadgets) are hosted within an [Apache Shindig 2.0](#)-based container. For further reading on gadget development, see the Shindig website or the [Google gadgets API reference site](#).

Gadgets exist in the browser within an IFrame. Therefore, gadgets may not directly call anything outside of their frame but instead must communicate using the `postMessage` Javascript function. The SocialMiner gadget container waits for certain messages on this mechanism before displaying user messages or closing reply templates.

In addition to IFrame limitations, gadgets are subject to standard same-domain request policies, meaning that AJAX POST, PUT, and DELETE requests may only be made to the host that originally served the gadget web page (in our case, the SocialMiner server). In practice, this means that all API requests must use the `makeRequest` function provided by the gadgets API to make gadget container-proxied requests for REST service URLs.

## Javascript Files

All of the objects and functions are found in `ccp-base.js`. This file is available from a SocialMiner server at `http://<server name or IP>/templates/reply/js/ccp-base.js`. Note that anything not documented in the API reference may be modified in the future.

## Getting Started

A basic example custom reply gadget is provided on the SocialMiner server at `http://<server name or IP>/templates/reply/custom_reply_sample.jsp`. Developers may download and experiment with this example to understand basic template structure and use of the `CcpSession` object.

## SocialMiner Javascript Objects

Documentation of SocialMiner Javascript APIs can be found on [Cisco DevNet](#) under Tools and Samples.

### CcpSession

The `CcpSession` object facilitates interaction with the reply template gadget container. The first thing a reply template gadget does is to create this object using the gadget page URL.

### APIMessage

Reply template gadgets interact with any web service that the server communicates with by using the [gadgets.io.makeRequest](#) function. The `APIMessage` objects provide a convenient way of calling APIs that invoke long-running operations (operations that don't immediately return a 200 on success, but instead return 201 or 202). `APIMessages` poll an operation until it is complete and only then call any provided callback.