Cisco SocialMiner Developer Guide, Release 11.6(2)

First Published: 2018-08-08

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: 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. (1721R)

© 2018 Cisco Systems, Inc. All rights reserved.

CONTENTS

PREFACE

Preface xiii

Change History xiii

About this Guide xiii

Audience xiii

Related Documents xiv

Obtaining Documentation and Submitting a Service Request xiv

Documentation Feedback xiv

CHAPTER 1

API Conventions 1

ID Variables 2

changeStamp 3

Passwords 3

HTTPS Responses 4

API Authentication 5

Field Constraints and Limitations 5

Cross Origin Resource Sharing (CORS) 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 Bubble Chat 13
GET (Chat Config) 13
Bubble Chat API Parameter 15

CHAPTER 5 Campaign 17
Campaign API Commands 17
POST 17
PUT 18
DELETE 19
GET 19
GET (List) 20
GET (Suggested Tags) 22
Campaign API Parameters 22

CHAPTER 6 Campaign Results 25
Campaign Results API Commands 25
GET 25
Campaign Results URL Parameters 26
Campaign Results Response Parameters 28

CHAPTER 7 Campaign Results Count 31
Campaign Results Count API Commands 31
GET 31

CHAPTER 8 GET (Chat Gateway Configuration) 33
PUT (Chat Gateway Configuration) 36
POST (Chat Gateway Configuration) 39

CHAPTER 9 Chat Feed 41

Create a Chat Feed 41
POST 41
Chat Request Form Submission 42
GET Method 43
GET (Create Chat Request) 43
Chat Feed API Parameters (Create Chat Request) 44
GET (Events) 44
GET (Events) Parameters 45
GET (Chat Transcript) 46
PUT (Update) 48
PUT (Leave Chat) 49
Put (Events) 50
DELETE 50
Chat Session Timeout 50

CHAPTER 10 **Chat Feedback** 51
Post 51
Chat Feedback Parameters 52

CHAPTER 11 **CCE Multichannel Routing** 53
Configuration for Multichannel Routing 53
Configuration for Multichannel Routing API Commands 53
GET 53
PUT 54
Callback 54
Callback API Commands 55
POST 55
GET (Create Callback Request) 56
GET 56
DELETE 57
Callback API Parameters 57
Task 58
Task Submission API Commands 59
POST 59

GET (Create Task Request) 60
GET 60
DELETE 61
Task Submission API Parameters 61
Task Resubmission API Commands 63
PUT 63
Task Resubmission API Parameters 63
Task Closure API Commands 64
PUT 64
Variables Used for Callback and Task APIs 64
Contact States and Reason Codes 68
XMPP Eventing Versus Task API Polling 70

CHAPTER 12 **Context Service** 73
Context Service API Commands 73
GET 73
PUT 74
Re-Initialize Context Service 74
PUT 74
Context Service Configuration API Parameters 75

CHAPTER 13 **Email** 77
Email API Commands 77
GET 77
PUT 78
Email API Parameters 78

CHAPTER 14 **Email Reply** 81
Email Reply API Commands 81
GET (Email) 81
GET (Email Reply Draft Data) 82
POST (Create Email Reply) 83
POST (Create Email Forward) 84
PUT (Email Draft) 84

Delete (Email Draft) 85
Email Reply API Parameters 85

CHAPTER 15 IMAP Folder Configuration 87
IMAP Folder Configuration 87
GET 87
PUT 88
IMAP Folder Configuration Parameters 88

CHAPTER 16 Feed API Commands 89
POST 89
POST (Create an IMAP Email Feed) 90
DELETE 91
GET (List) 91
GET 93
PUT 94
Feed API Parameters 94
Email Feed API Parameters 97
Email Feed Limitations 99

CHAPTER 17 Filter 101
Filter API Commands 101
Post 101
Delete 102
Get List 102
Get 103
GET (Script Content) 104
Put 104
Filter API Parameters 105
About Script Filters 106
Script Binding 106
Develop and Test Script Filters 107
Script Filter Security 110
Sample Script Filters 111

Script Filter for Social Contact Modification 111

CHAPTER 18 **Filter Results** 113

Filter Results API Commands 113

GET 113

CHAPTER 19 **Notification Rule** 115

Notification API Commands 116

POST 116

PUT 118

DELETE 118

GET (List) 118

GET 119

Notification API Parameters 120

Notification Keywords in Email and IM 122

HTTPS Notifications 122

CHAPTER 20 **Predefined Response** 125

Create Predefined Response 125

Get Predefined Response 127

List Predefined Response 128

List Predefined Response (By Group) 129

List Predefined Response (By Type) 130

List Predefined Response(Sorted by Title) 130

Update Predefined Response 130

Delete Predefined Response 131

CHAPTER 21 **Proxy** 133

Proxy API Commands 133

GET 133

PUT 134

Proxy API Parameters 134

CHAPTER 22 **Public URL Prefix for Chat Invitation** 135

GET 135
PUT 135

CHAPTER 23 Purge 137
Purge API Commands 137
GET (List) 137
PUT 138
Purge API Parameters 138

CHAPTER 24 Push Feed 139
Push Feed API Commands 139
POST 139
GET 139

CHAPTER 25 Reply Template API Commands 141
POST 141
DELETE 141
GET (List) 142
GET 143
PUT 143
Reply Template API Parameters 143

CHAPTER 26 Reporting Server 145
Reporting Server API Commands 145
GET 145
Reporting Server API Parameters 146

CHAPTER 27 Reporting User 147
Reporting User API Commands 147
POST 147
DELETE 148
GET (List) 148
GET 148

PUT 149
Reporting User API Parameters 149

CHAPTER 28 Serviceability 151
Serviceability API Commands 151
GET 151
GET (List) 152
Get (Performance Information) 155
Get (Context Service Status) 158
PUT (Triggers notification of the specified email contact in the specified campaign) 159
POST 160
Serviceability API Parameters 161
Feed statusDescription Values 172

CHAPTER 29 Social Contact 173
Social Contact API Commands 173
POST 173
GET 174
PUT (update) 176
PUT (Requeue Email) 177
GET (Chat Transcript) 179
GET (Search) 180
Social Contact API Parameters 183

CHAPTER 30 Socks Proxy 189
Socks Proxy API Commands 189
GET 189
PUT 190
Socks Proxy API Parameters 190

CHAPTER 31 Tag 191
Tag API Command 191
GET (List) 191

CHAPTER 32 URL Shortener 193
URL Shortener API Commands 193
POST 193
GET 194
GET (List) 195
PUT (update) 196
DELETE 196

CHAPTER 33 XMPP 197
XMPP API Commands 197
GET 197
PUT 198
XMPP API Parameters 198

APPENDIX A Reporting Database Connection 199
Reporting Database SQL Connection 199
Reporting Database Schema 199

APPENDIX B SocialMiner Server Configuration 205
Security Configuration Options 205
Port-Forward Firewall Configuration 205
Reverse Proxy 205

APPENDIX C XMPP BOSH Eventing 207
Publish and Subscribe 207
Nodes 208
ccp.campaign.updates 208
ccp.serviceability.eventingInfo 208
Events 208

APPENDIX D Custom reply Templates 213
Javascript Concepts 213

Preface

- Change History, on page xiii
- About this Guide, on page xiii
- Audience, on page xiii
- Related Documents, on page xiv
- Obtaining Documentation and Submitting a Service Request, on page xiv
- Documentation Feedback, on page xiv

Change History

Change	See	Date
Initial release of document for Release 11.6(2)		August 2018
Added Chat Gateway Configuration API		August 2018
Updated Serviceability API		August 2018

About this Guide

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

Audience

This document is intended for developers who wish to implement the REST API functions. The audience is assumed to be familiar with REST API Web services and the HTTPS protocol.

Related Documents

Document or Resource	Link
Cisco SocialMiner Documentation Guide	https://www.cisco.com/c/en/us/support/customer-collaboration/unified-contact-center-express/products-documentation-roadmaps-list.html
cisco.com site for Cisco SocialMiner documentation	https://www.cisco.com/c/en/us/support/customer-collaboration/socialminer/tsd-products-support-series-home.html
cisco.com site for Cisco Unified Contact Center Express documentation	https://www.cisco.com/c/en/us/support/customer-collaboration/unified-contact-center-express/tsd-products-support-series-home.html

Obtaining Documentation and Submitting a Service Request

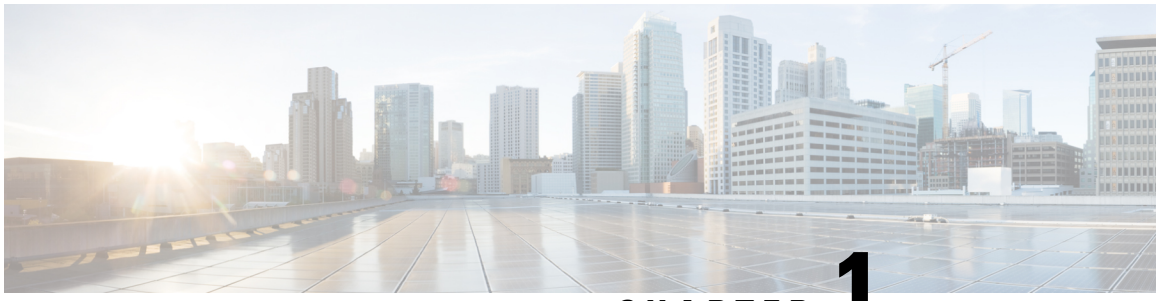
For information on obtaining documentation, using the Cisco Bug Search Tool (BST), submitting a service request, and gathering additional information, see What's New in Cisco Product Documentation.

To receive new and revised Cisco technical content directly to your desktop, you can subscribe to the . RSS feeds are a free service.

Documentation Feedback

To provide comments about this document, send an email message to the following address: contactcenterproducts_docfeedback@cisco.com

We appreciate your comments.



CHAPTER 1

API Conventions

SocialMiner uses REST-based API functions accessed over https. Five API functions are supported; each is mapped to an https operation or command.

Note HTTP and HTTPS protocols are the supported.

Not all functions are used for all components.

The URL format is:

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

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

The functions are:

- create (https 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:

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

  The id for the feed is *100162*.

  You can use this URL reference to retrieve the object with an https GET.

  Note In some APIs (for example, the callback API), you can also create objects with an https GET (create).

  Although GET (create) does not take a payload, the API developer must supply the required parameters in the URL.

- delete (https DELETE)—Deletes an object.
- get (https GET)—Returns data for an object. For objects for which multiple records exist, 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 (https 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 (https GET)—For objects for which multiple records can exist, 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 https 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, on page 2
- changeStamp, on page 3
- Passwords, on page 3
- HTTPS Responses, on page 4
- API Authentication, on page 5
- Field Constraints and Limitations, on page 5
- Cross Origin Resource Sharing (CORS), on 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 https 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: *https://<ServerIP>:<Port>/ccp-webapp/ccp/filter/<id>/results*.
- Use this URL to delete a feed: *https://<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.

publicId

Using https 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: *https://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/<publicId>/count*).
- Use this URL to update the configuration of a given campaign: *https://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/<publicId>*.

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

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. You must include the changeStamp 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 succeeds, the database increments the changeStamp by 1.

Passwords

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

HTTPS 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.
- **302 Found**: The requested resource resides temporarily under a different URI.
- **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.
- **403 Forbidden**: The operation is forbidden.
- **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.
- **408 Request Timeout**: The client did not produce a request within the time that the server was prepared to wait.
- **412 Precondition Failed**: The precondition given in one or more of the request-header fields evaluated to false when it was tested on the server.
- **415 Unsupported Media Type**: The server is refusing to service the request because the entity of the request is in a format not supported by the requested resource for the requested method.
- **500 Internal Server Error**: The server encountered an unexpected condition which prevented it from fulfilling the request.
- **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.

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 https basic authentication for those APIs that require it.

When you submit an API call through a web browser, for example *https://<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
https://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).

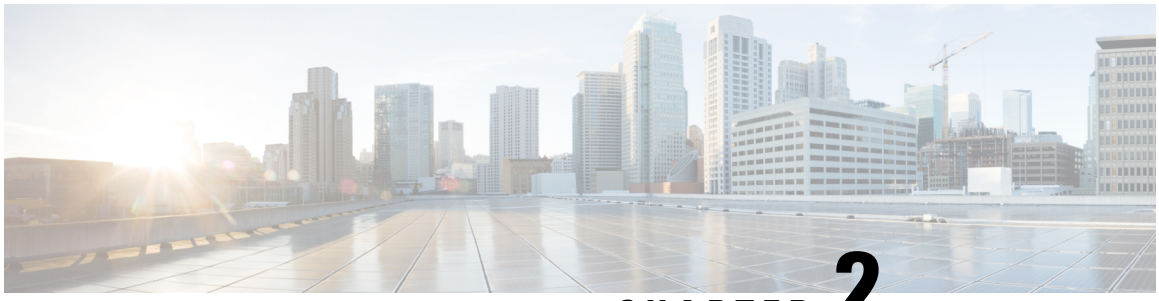
See Provisioning in the *SocialMiner User Guide* for information on limitations.

Cross Origin Resource Sharing (CORS)

Cross Origin Resource Sharing (CORS) refers to the accessibility of restricted resources on a web page requested from an external domain outside of its origin.

SocialMiner public APIs (Chat, Task etc.) support CORS. This implies that SocialMiner APIs can now directly be invoked via web browsers from an external site hosted on a different host and domain location than SocialMiner.

CORS support is permanently enabled, and pre-flight OPTIONS requests are also supported.



CHAPTER 2

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 installation can use this API.

- Authentication API Commands, on page 7
- Enable SSL for Active Directory Authentication, on 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.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/authentication/
HTTPS method:	GET

Example XML response:	`<Authentication>`
	`<enabled>true</enabled>`
	`<managerDistinguishedName>`
	`CN=administrator,CN=users,DC=ccbu-doc-ad,`
	`DC=cisco,DC=com`
	`</managerDistinguishedName>`

	<managerPassword>********</managerPassword> <primaryHost>10.xx.yyy.zzz</primaryHost> <primaryPort>3268</primaryPort> <primaryUseSSL>false</primaryUseSSL> <refURL> https://<ServerIP>:<Port>/ccp-webapp/ccp/ authentication </refURL> <roleName></roleName> </Authentication>
Parameters:	See Authentication API Parameters, on page 8.

PUT

Updates the authentication information from SocialMiner.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/authentication/
HTTPS method:	PUT
Example XML payload:	<Authentication> <enabled>true</enabled> <primaryHost>ad.server</primaryHost> <primaryPort>3268</primaryPort> <primaryUseSSL>false</primaryUseSSL> <managerDistinguishedName> cn=admin,ou=users,dc=ad,dc=server </managerDistinguishedName> <managerPassword>password</managerPassword> <roleName>CCP_Users</roleName> </Authentication>
Parameters:	See Authentication API Parameters, on page 8.

Authentication API Parameters

Parameters are optional unless otherwise noted.

Parameter	Description	Notes
enabled	True/False. Indicates if the authentication settings are used when trying to authenticate a user.	If this parameter is false, only the application administrator can access the system.
managerDistinguishedName	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.
managerPassword	The password of the user specified in the managerDistinguishedName field.	Required if the enabled parameter is true.

Parameter	Description	Notes
primaryPort	The host port.	Required if the enabled parameter is true.
primaryHost	The host address of the AD server.	Required if the enabled parameter is true.
primaryUseSSL	Indicates if a secure connection should be established.	If enabled, this parameter requires that a domain certificate is uploaded to the server and that the primaryPort allows secure connections. **If set to true, then you must follow the instructions in** Enable SSL for Active Directory Authentication, on page 9.
roleName	The name of an AD role or group.	All users in this AD role or group can access SocialMiner. Users in AD who are not members of this role or group cannot access SocialMiner. Blank or * indicates that all users in AD can 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 Enter the URL `https://<servername>/cmplatform` or use the Platform Administration link in the System Administration panel to open the Cisco Unified Operating System Administration page.

Step 2 Select **Security > Certificate Management**.

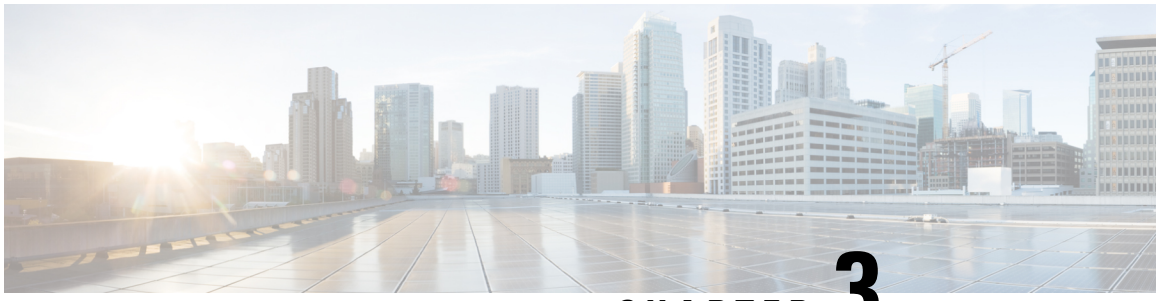
Step 3 Click **Upload Certificate**.

Step 4 For the Certificate Name, select **tomcat-trust**.

Step 5 In the Upload File field, click **Browse** and locate the file to upload. Select the certificate file you saved from the Active Directory server.

Step 6 Click **Upload File**.

Step 7 Run the CLI command `utils service restart Cisco Tomcat` to restart the Cisco Tomcat service.



CHAPTER 3

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, on 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 runs.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/filter/<objectId>/train
HTTPS method:	PUT

Parameters:	• **document**: Required if socialContact is not specified. String. The text on which to train the filter.
	• **socialContact**: Required if document is not specified. String. The URL of the social contact containing the content on which to train the filter.
	• **match**: Required. Boolean.
	  • Set to "True" to include social contacts with similar content in campaigns where the filter is applied.
	  • Set to "False" to exclude social contacts with similar content in campaigns where the filter is applied.

Example XML request payload using socialContact:	<TrainingRequest> <socialContact> https://[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/ B83B18F4100001292B3D088D0A568DDE </socialContact> <match>True</match> </TrainingRequest>
Example XML request payload using document:	<TrainingRequest> <document>This is very positive. I really like it. Performance was excellent. Great product.</document> <match>True</match> </TrainingRequest>
HTTPS response headers:	https/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:

Related Topics

ID Variables, on page 2

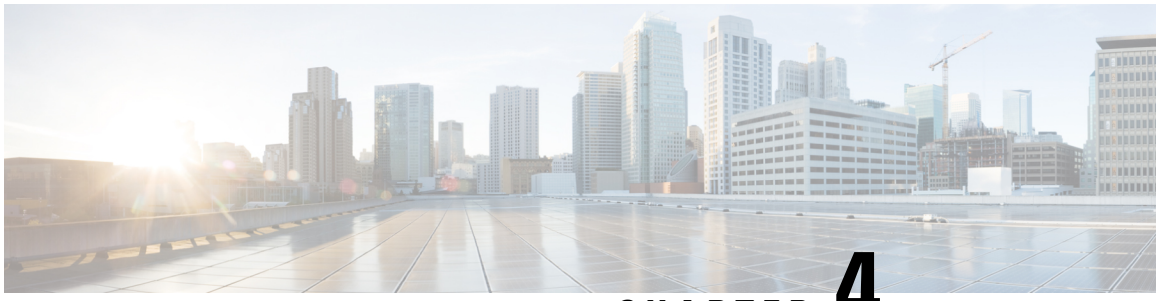
DELETE

Deletes all training data for a filter.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/filter/<objectId>/trainingdata
HTTPS method:	DELETE
HTTPS response headers:	https/1.1 200 OK Content-Type: text/plain Content-Length: 0 Date: Mon, 14 Jun 2010 14:22:30 GMT

Related Topics

ID Variables, on page 2



CHAPTER 4

Bubble Chat

The Bubble Chat API retrieves the bubble chat configuration details that are already configured in Unified CCX.

- GET (Chat Config), on page 13
- Bubble Chat API Parameter, on page 15

GET (Chat Config)

URL:	https://<SM_FQDN>/ccp/bubblechat?<query parameter>
HTTPS method:	GET
Input/Output format:	application/json
Parameter	See Bubble Chat API Parameter, on page 15.

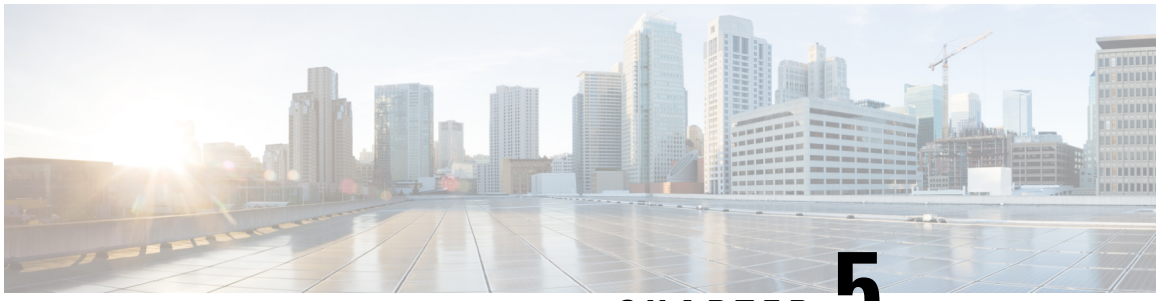
Example JSON response:	`GET /ccp/bubblechat?wid=1 HTTPS/1.1`
	`Host: SM-FQDN Accepts: application/json`
	
	`# Response`
	`Status: HTTPS/1.1 200 OK`
	`Payload:`
	`{`
	`"messages": {`
	`"agentJoinTimeoutMsg": "All customer care representatives are busy. Please wait or try again later.",`
	`"chatErrorMsg": "Sorry, the chat service is currently not available. Please try again later.",`
	`"offHourMessage": "Sorry, we are not available at the moment.",`
	`"welcomeMessage": "Welcome. Please wait while we connect you to a customer care representative."`
	`},`
	`"proactiveChat": false,`
	`"contextServiceFieldSets":`
	`"com.ccx-social_miner.pod,com.ccx-media_sense.pod",`
	`"chatForm": {`
	`"problemStatements": {`
	`"statements": [`
	`{`
	`"statement": "PL problem statement",`
	`"csqTag": "Chat_Csq3"`
	`},`
	`{`
	`"statement": "HL problem statement",`
	`"csqTag": "Chat_Csq2"`
	`}`
	`],`
	`"caption": "Choose a Problem Statement"`
	`},`
	`"bubbleStyle": {`
	`"titleText": "chat bubble 3",`
	`"titleTextColor": "#1f1f1F",`
	`"buttonText": "chat button 3",`
	`"buttonTextColor": "#1f1f1F",`
	`"buttonBackgroundColor": "#ebebec",`
	`"agentMessageTextColor": "#ebebec",`
	`"agentMessageBackgroundColor": "#1f1f1F",`
	`"fontFace": "Arial"`
	`},`
	`"formFields": [`
	`"Name",`
	`"Details",`
	`"Email",`
	`"PhoneNumber",`
	`"CustomForm"`
	`]

HTTPS response headers:	200 OK 400 Bad Request 404 Not Found 405 Method Not Allowed 500 Internal Server error - due to various internal failure scenarios See HTTPS Responses, on page 4 for more information about the response codes.

Bubble Chat API Parameter

The following table lists the required query parameter for the Bubble Chat API.

Parameter	Description	Notes
wid	A unique ID for each chat widget.	Integer. Required for GET (Chat config).



CHAPTER 5

Campaign

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

Campaigns are collections of feeds (see feeds) 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, on page 17

Campaign API Commands

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

Related Topics

POST, on page 17
PUT, on page 18
DELETE, on page 19
GET, on page 19
GET (List), on page 20
GET (Suggested Tags), on page 22
Campaign API Parameters, on page 22

POST

Creates a campaign.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/campaign
HTTPS method:	POST
Parameters:	See Campaign API Parameters, on page 22.
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>

	<pre>https://[ServerIP]:[Port]/ccp-webapp/ccp/feed/5000
	</chatInvitationFeed>
	<feeds>
	<feed>
	https://[ServerIP]:[Port]/ccp-webapp/ccp/feed/5000
	</feed>
	<feed>
	https://[ServerIP]:[Port]/ccp-webapp/ccp/feed/5001
	</feed>
	</feeds>
	<filters>
	<filter>
	https://[ServerIP]:[Port]/ccp-webapp/ccp/filter/6000
	</filter>
	<filter>
	https://[ServerIP]:[Port]/ccp-webapp/ccp/filter/6001
	</filter>
	</filters>
	</Campaign></pre>

HTTPS response headers:	If the campaign is successfully created, the URL of the created resource is returned.
	<pre>https/1.1 201 Created
	Location: https://<ServerIP>:<Port>/ccp-webapp/ccp/
	campaign/MyTestCampaign
	Content-Type: text/plain
	Content-Length: 0
	Date: Tue, 12 Jan 2010 16:41:14 GMT</pre>
	See also HTTPS Responses.

PUT

Updates an existing campaign.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/<publicId> For more information about <publicId>, see ID Variables, on page 2.
HTTPS method:	PUT
Parameters:	See Campaign API Parameters, on page 22.

Example XML request payload:	<pre><Campaign>
	<name>MyTestCampaign</name>
	<publicid>MyTestCampaign</publicid>
	<description>This is my test campaign</description>
	<includeExpr>Cisco Expert Advisor</includeExpr>
	<excludeExpr>ICM</excludeExpr>
	<chatInvitationFeed>
	https://[ServerIP]:[Port]/ccp-webapp/ccp/feed/5000
	</chatInvitationFeed>
	<feeds>
	<feed>
	https://[ServerIP]:[Port]/ccp-webapp/ccp/feed/5000
	</feed>
	<feed>
	https://[ServerIP]:[Port]/ccp-webapp/ccp/feed/5001
	</feed>
	</feeds>
	<filters></pre>

```
    <filter>
     https://[ServerIP]:[Port]/ccp-webapp/ccp/filter/6000
    </filter>
    <filter>
     https://[ServerIP]:[Port]/ccp-webapp/ccp/filter/6001
    </filter>
  </filters>
  <changeStamp>8</changeStamp>
</Campaign>
```

DELETE

Deletes an existing campaign.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/<publicId> For more information about <publicId>, see ID Variables, on page 2.
HTTPS method:	DELETE

HTTPS response headers:	`https/1.1 200 OK`
	`Content-Type: text/plain`
	`Content-Length: 0`
	`Date: Tue, 12 Jan 2010 17:03:54 GMT`

GET

Returns the data for a single campaign.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/<publicId>?metrics=<true/false> For more information about <publicId>, see ID Variables, on page 2.
HTTPS method:	GET
Parameters:	See Campaign API Parameters, on page 22.

Example XML response:	**Note** The“ refURL” is a copy of the URL requested.
	
	`<Campaign>`
	`  <refURL>`
	`  https://[ServerIP]:[Port]/ccp-webapp/ccp/`
	`  campaign/MyTestCampaign`
	`  </refURL>`
	`  <resultsURL>`
	`  https://[ServerIP]:[Port]/ccp-webapp/ccp/`
	`  campaign/MyTestCampaign/results`
	`  </resultsURL>`
	`  <suggestedTagsURL>`
	`  https://[ServerIP]:[Port]/ccp-webapp/ccp/`
	`  campaign/MyTestCampaign/suggestedtags`
	`  </suggestedTagsURL>`
	`  <publicid>MyTestCampaign</publicid>`
	`  <description>This is my test campaign</description>`
	`  <includeExpr>Cisco Expert Advisor</includeExpr>`
	`  <excludeExpr>ICM</excludeExpr>`
	`  <chatInvitationFeed>`

	<pre>https://[ServerIP]:[Port]/ccp-webapp/ccp/feed/5000
	</chatInvitationFeed>
	<changeStamp>12345</changeStamp>
	<feeds>
	<feed>
	https://[ServerIP]:[Port]/ccp-webapp/ccp/feed/5000
	</feed>
	<feed>
	https://[ServerIP]:[Port]/ccp-webapp/ccp/feed/5001
	</feed>
	</feeds>
	<filters>
	<filter>
	https://[ServerIP]:[Port]/ccp-webapp/ccp/filter/6000
	</filter>
	<filter>
	https://[ServerIP]:[Port]/ccp-webapp/ccp/filter/6001
	</filter>
	</filters>
	<metrics>
	<socialContactCount>12</socialContactCount>
	</metrics>
	</Campaign></pre>
	The metrics element contains <socialContactCount>, which is the number of social contacts associated with this campaign .

HTTPS response headers:	<pre>https/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.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/campaign?summary=<true/false> Where *summary* is an optional query parameter that is false by default. OR https://<ServerIP>:<Port>/ccp-webapp/ccp/campaign?metrics=<true/false> Where *metrics* is an optional query parameter that is false by default.
HTTPS method:	GET

Example XML responses:	When summary is true and metrics is false:
	
	<pre><Campaigns>
	<Campaign>
	<refURL>
	https://[ServerIP]:[Port]/ccp-webapp/ccp/
	campaign/MyTestCampaign
	</refURL>
	</Campaign>
	<Campaign>
	<refURL></pre>

```
      https://[ServerIP]:[Port]/ccp-webapp/ccp/
      campaign/MyTestCampaign2
    </refURL>
  </Campaign>
</Campaigns>
```

When summary is false and metrics is false:

```
<Campaigns>
  <Campaign>
    <refURL>
      https://[ServerIP]:[Port]/ccp-webapp/ccp/
      campaign/MyTestCampaign
    </refURL>
    <name>MyTestCampaign</name>
    <publicid>MyTestCampaign</publicid>
    <description>This is my test campaign</description>
    ...
  </Campaign>
  <Campaign>
    <refURL>
      https://[ServerIP]:[Port]/ccp-webapp/ccp/
      campaign/MyTestCampaign2
    </refURL>
    <name>MyTestCampaign2</name>
    <publicid>MyTestCampaign2</publicid>
    <description>This is my test campaign</description>
    ...
  </Campaign>
</Campaigns>
```

When summary is false and metrics is true:

```
<Campaigns>
  <Campaign>
    <refURL>
      https://[ServerIP]:[Port]/ccp-webapp/ccp/
      campaign/MyTestCampaign
    </refURL>
    <name>MyTestCampaign</name>
    <publicid>MyTestCampaign</publicid>
    <description>This is my test campaign</description>
    <metrics>
      <socialContactCount>12</socialContactCount>
    </metrics>
    ...
  </Campaign>
  <Campaign>
    <refURL>
      https://[ServerIP]:[Port]/ccp-webapp/ccp/
      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.

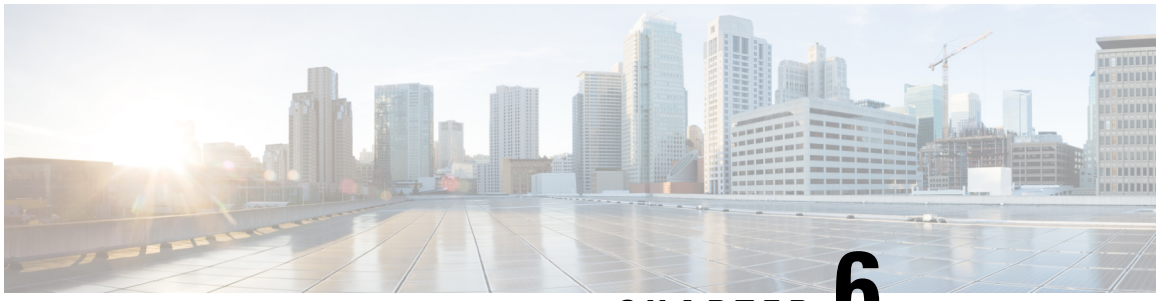
URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/<publicId>/suggestedtags For more information about <publicId>, see ID Variables, on page 2.
HTTPS method:	GET
Example XML response:	The first 10 suggested tags are returned. <SuggestedTags> <tags> <tag>tag1</tag> <tag>tag2</tag> <tag>tag3</tag> </tags> </SuggestedTags>

Campaign API Parameters

Parameters are optional unless otherwise noted.

Parameter	Description	Notes
changeStamp	The change stamp of the campaign record.	Integer. Defaults to 0. Required for PUT. For more information, see changeStamp.
chatInvitationFeed	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 users to chat from the SocialMiner reply templates.
description	The description of the campaign.	
excludeExpr	The searching expression to exclude.	
feeds	A list of feeds linked to the campaign.	
filters	A list of filters linked to the campaign.	
includeExpr	The searching expression to include.	

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



CHAPTER 6

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, on page 25

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 25

Campaign Results URL Parameters, on page 26

Campaign Results Response Parameters, on page 28

GET

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

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/<publicId>/results For more information about <publicId>, see ID Variables, on page 2.
HTTPS method:	GET
URL parameters:	See Campaign Results URL Parameters, on page 26.
Example response:	<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/"> <title>My Chat Campaign </title><link rel="self" href="https://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/My_Chat_ Campaign/results?timestamp=1531413651109"/> <link rel="countsince" href="https://[ServerIP]:[Port] /ccp-webapp/ccp/campaign/My_Chat_Campaign/count?postId= 8F5CBBCD10000164000000920A4E5F26"/>

```
    <subtitle>This feed has been created by Cisco SocialMiner
     </subtitle><id>
https://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/My_Chat_Campaign</id>
      <updated>2018-07-12T16:40:51Z</updated>
     <dc:date>2018-07-12T16:40:51Z</dc:date>
     <entry>
     <title>Web Chat</title>
     <link rel="alternate" href="https://
[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/
8F5CBBCD10000164000000920A4E5F26"/>
<link rel="socialcontact" href="https://
[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/
8F5CBBCD10000164000000920A4E5F26"/>
<author><name>John Doe</name></author><id>
https://[ServerIP]:[Port]/ccp/socialcontact/
8F5CBBCD10000164000000920A4E5F26</id>
      <updated>2018-07-12T16:39:36Z</updated>
      <published>2018-07-12T16:39:36Z</published>
      <content type="application/xml">
      <ChatTranscript xmlns="">
      <id>8F5CBBCD10000164000000920A4E5F26</id>
      <startDate>1531413576639</startDate>
      <endDate>1531413634821</endDate>
      <chatInitiator>John Doe</chatInitiator>
      <agents><agent><agentName>sjefferson</agentName>
      <nickName>Sandra Jefferson</nickName>
       </agent>
       </agents>
       <transcript>...
       </transcript>
       </ChatTranscript>
       </content>
 <summary type="html">Re-issue tickets
 </summary>
       <dc:creator>John Doe</dc:creator>
       <dc:date>2018-07-12T16:39:36Z</dc:date>
       <ccp:scstatustimestamp>1531413625360</ccp:scstatustimestamp>
       <ccp:scstatus>handled</ccp:scstatus>
       <ccp:scstatusreason>chat_agent_ended</ccp:scstatusreason>
       <ccp:scstatususerid>admin</ccp:scstatususerid>
       <ccp:sourcetype>chat</ccp:sourcetype>
       <ccp:scissoftlocked>false</ccp:scissoftlocked>
 </entry>
 <entry>
 ...
 </entry>
 <entry>
 ...
 </entry>
 </feed>
```

For more information about the response parameters, see Campaign Results Response Parameters, on page 28.

Campaign Results URL Parameters

Parameters are optional unless otherwise noted.

Parameter	Description	Notes
filterStatus	Display contacts whose status matches a status within this field.	String. Defaults to all if the parameter is not specified. 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): • RESERVED • HANDLED • DRAFT • UNREAD • DISCARDED • QUEUED • Multiple status example: https://192.168.0.1/ccp-webapp/ccp / campaign/ Business_News/results **?filterStatus=RESERVED &** **filterStatus=HANDLED**
filterTag	Display contacts whose filters matches one or more of the tags in this field.	String. Defaults to all tags if not specified. Example: https://192.168.0.1/ccp-webapp/ccp /campaign/Business_News/results **?filterTag=tag1** **&filterTag=tag2**
includePostWithPostId	Specifies whether or not to display contacts with ID equal to postId.	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.
postId	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.

Parameter	Description	Notes
timestamp	A given time and date.	Integer. Displays results older than this timestamp. Defaults to the time of request if not provided. If startIndex is not specified, then timestamp assumes startIndex = 0.
resultsPerPage	The maximum number of results to be returned.	Integer. Default is 50 and maximum is 200.
startIndex	The number of results to skip based on the timestamp.	Integer. Used for pagination. 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 `feed/link rel="self"` and a startIndex of 50. Page 3 would use the same timestamp and a startIndex of 100, and so on.

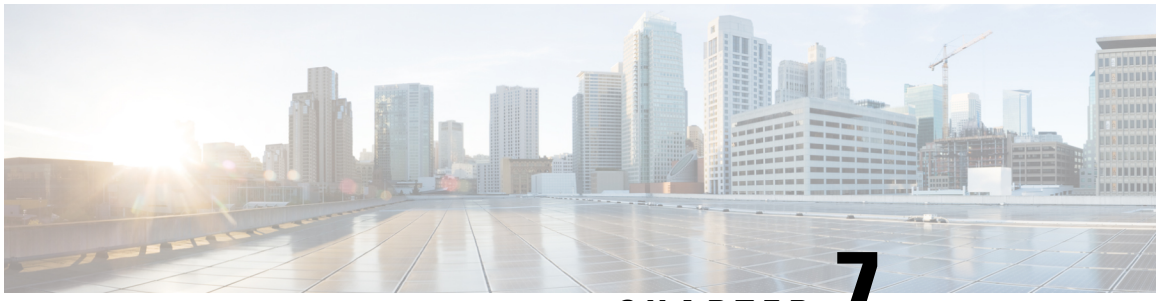
- If timestamp is provided and startIndex is not provided, then the results are displayed up to the "resultsPerPage" with a creation date older than "timestamp", starting at index 0.
- If timestamp is not provided and startIndex is provided, then the results are displayed up to the "resultsPerPage" with 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
feed/title	The name of the campaign.	
feed/link rel = self & feed/id	The URL of the results that were requested.	
feed/link rel = countsince	The URL for the API call for the number of new contacts since this result was retrieved.	See Campaign Results Count, on page 31.
feed/link rel = next	The URL to the next 50 results.	Present only when more than 50 results are left in the campaign. Represented in the SocialMiner user interface by the More button.
entry/title	The title of the contact.	
entry/link rel = alternate & entry/id	The URL to the contact.	

Parameter	Description	Notes
entry/link rel = socialcontact	The URL for the API call of this contact.	
entry/summary	The content of the contact.	
entry/published	The date and time that the contact was published. 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 *YYYY-MM-DD*T*HH:MM:SS*Z.
entry/ccp:statustimestamp	The timestamp of the last change to the status of the contact.	
entry/ccp:status	The current status of the contact.	
entry/ccp:scstatusreason	The reason the contact changed to its current status.	
entry/ccp:sctags/ccp:sctag	One or more tags associated with this contact.	
entry/ccp:scstatususerid	The last user to change the status of this contact. If blank and the status is unread, then this contact has never had a status change.	
entry/ccp:sourcetype	The type of feed that generated or fetched the contact.	
entry/ccp:scissoftlocked	Whether or not the contact state can be modified using the SocialMiner user interface.	



CHAPTER 7

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, on page 31

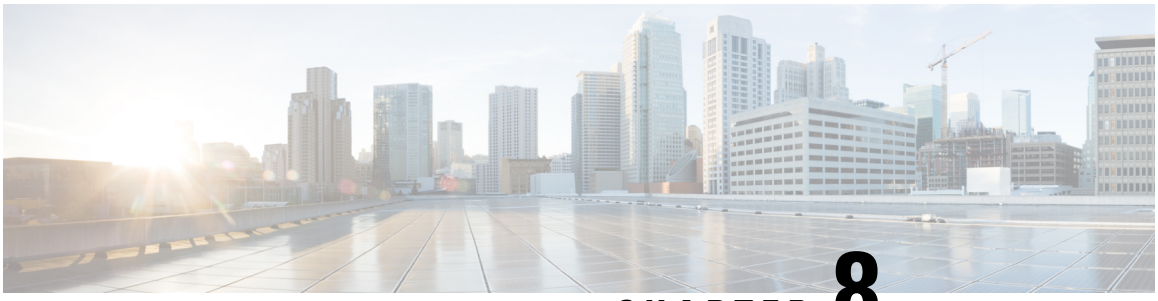
Campaign Results Count API Commands

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

GET

Gets the number of results in a campaign.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/campaign/ <publicId>/count For more information about <publicId>, see ID Variables, on page 2.
HTTPS method:	GET
URL parameter:	**postId**: 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, as shown in the following example: `<link rel="countsince" href="https://[ServerIP]:[Port]/ ccp-webapp/ccp/campaign/MyTestCampaign/count? postId=92517B6610000128295CEBB40A568DDE" />`
Example XML response:	`<count>44</count>`



CHAPTER 8

GET (Chat Gateway Configuration)

Retreives the chat gateway related information from SocialMiner.

URI:	https://<Server IP>:<Port>/ccp-webapp/ccp/chatGatewayConfig
HTTP Method:	GET
Request Headers	**Authorization:** Basic **Accept:** application/xml
Request Body	EMPTY
Response Status	200 OK
Response Headers	Content-Type: application/xml

Response Body	

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?>
<chatGatewayConfig>
  <channels>
    <channel>
      <auth>

<pageAccessToken>my_facebook_page_access_token</pageAccessToken>


<verificationToken>my_facebook_verify_token</verificationToken>

      </auth>
      <chatFeedId>100000</chatFeedId>
      <enabled>true</enabled>
      <ratingEnabled>true</ratingEnabled>
      <messages>
        <entry>
          <key>CHAT_ENDED</key>
          <value>Your chat has ended. Thank
you.</value>
        </entry>
        <entry>
          <key>UNSUPPORTED_MESSAGE</key>
          <value>Sorry, we support only text
 messages.</value>
        </entry>
        <entry>
          <key>RATING_OFFER</key>
          <value>Would you like to rate your
 chat experience on a scale of 1 [worst] to
5 [best]? Select a rating.</value>
        </entry>
        <entry>
         <key>PROBLEM_STATEMENT_CAPTION</key>

          <value>How can we help you? Choose
 from one of the options.</value>
        </entry>
        <entry>
          <key>CHAT_ENDED_UNKNOWN_ERROR</key>

         <value>Oops! An error occurred. Chat
 has ended. Try again later.</value>
        </entry>
        <entry>
          <key>WAIT_FOR_AGENT</key>
          <value>Just a moment...finding an
expert to chat with you.</value>
        </entry>
        <entry>
          <key>AGENT_JOIN_TIMEOUT</key>
          <value>We are busy at the moment.
You can continue to wait or try again
later.</value>
        </entry>
        <entry>
          <key>WELCOME_MESSAGE</key>
          <value>Hello! Welcome to ABCD
Customer Care.</value>
        </entry>
        <entry>
          <key>CHAT_ENDED_INACTIVITY</key>
          <value>Chat has ended due to
```

	inactivity. You can contact us again.</value> </entry> <entry> <key>CHAT_ENDED_AGENT_UNAVAILABLE</key> <value>Sorry, we could not find an expert to chat with you at this moment. Try again later.</value> </entry> <entry> <key>RATING_COMPLETE</key> <value>Thank you for the feedback.</value> </entry> </messages> <problemStatements> <problemStatement> <queueId>1</queueId> <title>Warranty Issue</title> </problemStatement> <problemStatement> <queueId>2</queueId> <title>Sales Enquiry</title> </problemStatement> <problemStatement> <queueId>3</queueId> <title>Complaints</title> </problemStatement> </problemStatements> <type>FACEBOOK_MESSENGER</type> </channel> </channels> <httpProxy> <enabled>true</enabled> <host>my-http-proxy.example.com</host> <port>8080</port> </httpProxy> <version>1.0</version> </chatGatewayConfig>

- PUT (Chat Gateway Configuration), on page 36
- POST (Chat Gateway Configuration), on page 39

PUT (Chat Gateway Configuration)

Updates the chat gateway related configuration settings.

URI:	https://<Server IP>:<Port>/ccp-webapp/ccp/chatGatewayConfig
HTTP Method:	PUT
Request Headers	**Authorization:** Basic **Content-Type:** application/xml

Request Body	

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?>
<chatGatewayConfig>
  <channels>
    <channel>
      <auth>

<pageAccessToken>my_facebook_page_access_token</pageAccessToken>


<verificationToken>my_facebook_verify_token</verificationToken>

      </auth>
      <chatFeedId>100000</chatFeedId>
      <enabled>true</enabled>
      <ratingEnabled>true</ratingEnabled>
      <messages>
        <entry>
          <key>CHAT_ENDED</key>
          <value>Your chat has ended. Thank
you.</value>
        </entry>
        <entry>
          <key>UNSUPPORTED_MESSAGE</key>
          <value>Sorry, we support only text
 messages.</value>
        </entry>
        <entry>
          <key>RATING_OFFER</key>
          <value>Would you like to rate your
 chat experience on a scale of 1 [worst] to
5 [best]? Select a rating.</value>
        </entry>
        <entry>
         <key>PROBLEM_STATEMENT_CAPTION</key>

          <value>How can we help you? Choose
 from one of the options.</value>
        </entry>
        <entry>
          <key>CHAT_ENDED_UNKNOWN_ERROR</key>

         <value>Oops! An error occurred. Chat
 has ended. Try again later.</value>
        </entry>
        <entry>
          <key>WAIT_FOR_AGENT</key>
          <value>Just a moment...finding an
expert to chat with you.</value>
        </entry>
        <entry>
          <key>AGENT_JOIN_TIMEOUT</key>
          <value>We are busy at the moment.
You can continue to wait or try again
later.</value>
        </entry>
        <entry>
          <key>WELCOME_MESSAGE</key>
          <value>Hello! Welcome to ABCD
Customer Care.</value>
        </entry>
        <entry>
          <key>CHAT_ENDED_INACTIVITY</key>
          <value>Chat has ended due to
```

	<pre>inactivity. You can contact us again.</value>
	
	        </entry>
	        <entry>
	
	<key>CHAT_ENDED_AGENT_UNAVAILABLE</key>
	          <value>Sorry, we could not find an
	 expert to chat with you at this moment. Try
	 again later.</value>
	        </entry>
	        <entry>
	          <key>RATING_COMPLETE</key>
	          <value>Thank you for the
	feedback.</value>
	        </entry>
	      </messages>
	      <problemStatements>
	        <problemStatement>
	          <queueId>1</queueId>
	          <title>Warranty Issue</title>
	        </problemStatement>
	        <problemStatement>
	          <queueId>2</queueId>
	          <title>Sales Enquiry</title>
	        </problemStatement>
	        <problemStatement>
	          <queueId>3</queueId>
	          <title>Complaints</title>
	        </problemStatement>
	      </problemStatements>
	      <type>FACEBOOK_MESSENGER</type>
	    </channel>
	  </channels>
	  <httpProxy>
	    <enabled>true</enabled>
	    <host>my-http-proxy.example.com</host>
	    <port>8080</port>
	  </httpProxy>
	  <version>1.0</version>
	</chatGatewayConfig></pre>

Response Status	200 OK
Response Headers	Content-Type: application/xml
Response Body	EMPTY

POST (Chat Gateway Configuration)

Use this to test the chat gateway configuration settings done.

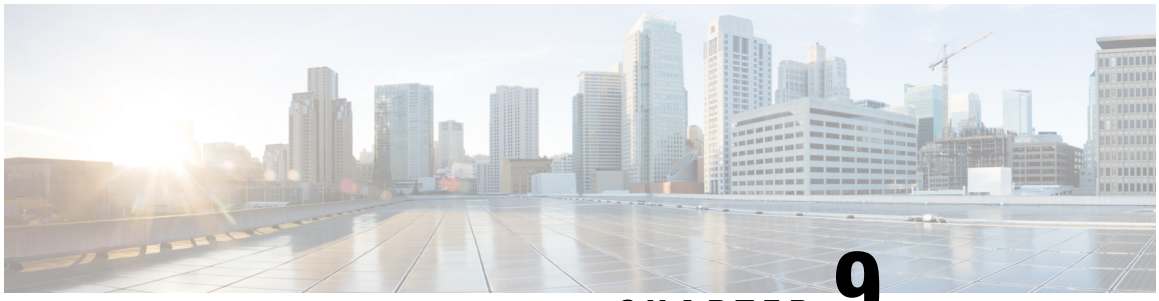
URI:	https://<SM_HOST>/ccp-webapp/ccp/chatGatewayConfig/test/facebook
HTTP Method:	POST

Request Headers	**Authorization:** Basic **Accept:** application/xml **Content-Type:** NONE

Request Body	`<fbmConfig>`
	`<accessToken>PAGE_ACCESS_TOKEN</accessToken>`
	`<httpProxy>`
	`<hostname>my-http-proxy.example.com</hostname>`
	`<port>8080</port>`
	`</httpProxy>`
	`</fbmConfig>`

Response Status	200 OK
Response Headers	Content-Type: application/xml

Response Body	`<FBMConfigTestStatus>`
	`<reason></reason>`
	`<status>PASSED</status>`
	`</FBMConfigTestStatus>`



CHAPTER 9

Chat Feed

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

For more information on feeds, see Feed.

- Create a Chat Feed, on page 41
- GET (Events), on page 44
- GET (Chat Transcript), on page 46
- PUT (Update), on page 48
- PUT (Leave Chat), on page 49
- Put (Events), on page 50
- DELETE, on page 50
- Chat Session Timeout, on page 50

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. Look in the location field of the https response header for the reference URL (refURL) of the newly-created feed.

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).

URL:	https://<ServerIP>:<Port>/ccp/chat
HTTPS method:	POST
Example XML request payload:	<SocialContact> <feedRefURL>https://[ServerIP]:[Port]/ ccp-webapp/ccp/feed/134268</feedRefURL> <author>Test_author</author> <title>Social contact title</title> </SocialContact>
The contact is returned in the location field in the header:	https://<ServerIP><:<Port>/ccp-webapp/ccp/socialcontact/ 6EEF968810000132000015F60A568DFB
The chat room is in the Extension Fields of the content response:	<SocialContact> <author>Test_author</author> <createdDate>1316121187977</createdDate> <description/> <extensionFields> <extensionField> <name>chatRoom</name> <value>socialminer_chat.3@conference.127.0.0.1</value> </extensionField> </extensionFields> <id>6EEF968810000132000015F60A568DFB</id> <link> https://[ServerIP]:[Port]/ ccp-webapp/ccp/socialcontact/ 6EEF968810000132000015F60A568DFB </link> <publishedDate>1316121187976</publishedDate> <refURL> https://[ServerIP]:[Port]/ ccp-webapp/ccp/socialcontact/ 6EEF968810000132000015F60A568DFB </refURL> <replyToId/> <status>unread</status> <statusTimestamp>1316121187976</statusTimestamp> <statusUserId/> <tags/> <title>Social contact title</title> </SocialContact>
Parameters:	See Chat Feed API Parameters (Create Chat Request), on page 44.
HTTPS response headers:	See HTTPS Responses.

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 https://<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 44.

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 https://<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.

URL:	https://<ServerIP>:<Port>/ccp/chat/<id>/redirect
HTTPS method:	GET
Example https request:	`https://<ServerIP>:<Port>/ccp/chat/<id>redirect?title=Test_Title&author=Test_Author`
Parameters:	See Chat Feed API Parameters (Create Chat Request), on page 44.
Response:	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 **https://<ServerIP>:<Port>/ccp/chat** (without the feedid).

Chat Feed API Parameters (Create Chat Request)

Parameters are optional unless otherwise noted.

Parameter	Description	Notes
title	The name of the contact.	Required
author	The author of the contact.	Required
description	A description of the content or context of the contact.	
tags	One or more tags that are associated with this contact.	
extensionField	A custom name/value pair.	The name is formatted (extensionField_<field name>). 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.

URL example:	https://<ServerIP>:<Port>/ccp/chat?eventid=0&all=false
HTTPS method:	GET
Parameter:	See below.
Example response payload:	<ChatEvents> <PresenceEvent> <id>2</id> <from>Steve</from> <status>joined</status> </PresenceEvent> <TypingEvent> <id>2</id> <from>Steve</from> <status>composing</status>

	<MessageEvent> <id>3</id> <from>Steve</from> <body>Hi There</body> </MessageEvent> <MessageEvent> <id>4 </id> <from>Steve</from> <body>How can I help you?</body> </MessageEvent> <TypingEvent> <id>3</id> <from>Steve</from> <status>paused</status> </TypingEvent> </ChatEvents>
HTTPS response headers:	200 (Succeeded) or 400/500 (Failed). See HTTPS Responses.

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: `https://<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: • id: sequential id of the event. • from: sender of the message. • body: URL-encoded body of the message.
PresenceEvent	A user joined or left the session: • id: sequential id of the event. • from: user who joined/left the room. • status: joined|left.
TypingEvent	• id: sequential id of the event. • from : user who joined the room. • status : indicates the typing status. It can either be "composing" or "paused". **Note** The Typing Status provided is case sensitive and has to either of the two as mentioned above. Any other value provided will not be accepted.

Event	Description and child events

StatusEvent	Chat session status:
	• id: sequential id of the event.
	• status: status information.
	  • chat_finished: the chat is finished with no problems (not currently used).
	  • chat_finished_error: the chat is finished due to an error; this is not recoverable (not currently used).
	  • chat_issue: there is an issue with the chat session; this may be recoverable.
	  • chat_ok: the chat session is ok again.
	• detail: If the XMPP Server fails, a StatusEvent will be sent with status set to *chat_issue* and detail set to *XMPP Server is down*.

GET (Chat Transcript)

XML Transcript Download

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

URL:	https://<ServerIP>:<Port>/ccp/chat/transcript.xml
HTTPS method:	GET
Example XML response:	<ChatTranscript><startDate>1389540103973</startDate> <transcript> <chat><time>1389540103973</time><name>Agent</name><msg>Hi</msg></chat> <chat><time>1389540108058</time><name>Dave</name><msg>Hello</msg></chat> <chat><time>1389540111001</time><name>Agent</name><msg>How may I help you?</msg></chat> <chat><time>1389540114026</time><name>Dave</name><msg>I have a query</msg></chat> </transcript> </ChatTranscript>
Elements:	**time**: timestamp of the chat in GMT. **name**: agent or customer name. **msg**: the chat message.
HTTPS response headers:	400 Bad Request 404 Not Found 501 Not Implemented See HTTPS Responses, on page 4 for more information about the response codes.

HTML Transcript Download

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

URL:	https://<serverIP>:<Port>/ccp/chat/transcript?locale=<locale>
HTTPS method:	GET
HTTPS response headers:	200 OK 400 Bad Request 404 Not Found See HTTPS Responses, on page 4 for more information about the response codes.

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

`ChatTranscript_<Customer_Name_spaces_replaced_by_underscores>-YYYY_DD_MMM_hh_mm.html`

For example, `ChatTranscript_John_Doe_2018_26_Jan_13_45.html`.

Note

- The time zone for all time values mentioned in the HTML 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:

  https://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
es_ES	Spanish
fi_FI	Finnish
fr_FR	French (France)
it_IT	Italian
nb_NO	Norwegian
nl_NL	Dutch
pl_PL	Polish

String	Language
pt_BR	Portuguese
ru_RU	Russian
sv_SE	Swedish
tr_TR	Turkish
zh_CN	Chinese (Simplified)
zh_TW	Chinese (Traditional)
ja_JP	Japanese
ko_KR	Korean

PDF Transcript Download (Not Used for Transcript Download - Used to Redirect to HTML Transcript Download API)

Since SocialMiner 11.6(2), this API is not used to download chat transcripts as PDF files. Requests to this API will get unconditionally redirected to the HTML transcript download API (see above).

URL:	https://<serverIP>:<Port>/ccp/chat/transcript.pdf?locale=<locale>
HTTPS method:	GET
HTTPS response headers:	301 Moved Permanently **Note** Since SocialMiner 11.6(2), requests to this API gets the response with the HTTP status 301 (MOVED PERMANENTLY) and the Location header of the response contains a complete URL to the HTML Transcript Download API . Most clients (including all browsers) will get automatically redirected to this URL in the Location header. This will avoid any impact to existing third party clients using this API.

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.

URL:	https://<ServerIP>:<Port>/ccp/chat
HTTPS method:	PUT
Example response payload:	<ChatEvents> <PresenceEvent> <id>1</id> <from>Steve</from> <status>joined</status> </Presence> <MessageEvent> <id>2</id> <from>Steve</from> <body>Hi There</body> </Message> <MessageEvent> <id>3</id> <from>Steve</from> <body>How can I help you?</body> </Message> </ChatEvents>
Responses:	200 (Succeeded) or 400/500 (Failed). See HTTPS Responses.

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.

URL:	https://<ServerIP>:<Port>/ccp/chat/leaveChat
HTTPS method:	PUT
HTTPS response headers:	200 OK 400 Bad Request For more information about the response codes, see HTTPS Responses, on page 4.

Put (Events)

This API sends a current chat typing status event to the chat room which is sent by the client.

URL:	https://<ServerIP>:<Port>/ccp/chat/event
HTTPS method:	PUT
Example response payload:	<TypingEvent> <from>Steve</from> <status>composing</status> </TypingEvent>
HTTPS response headers:	200 OK. 400 Bad Request. See HTTPS Responses.

Note The Typing Status values should either be "composing" or "paused". They are case-sensitive in nature. Any other values povided will not be accepted.

DELETE

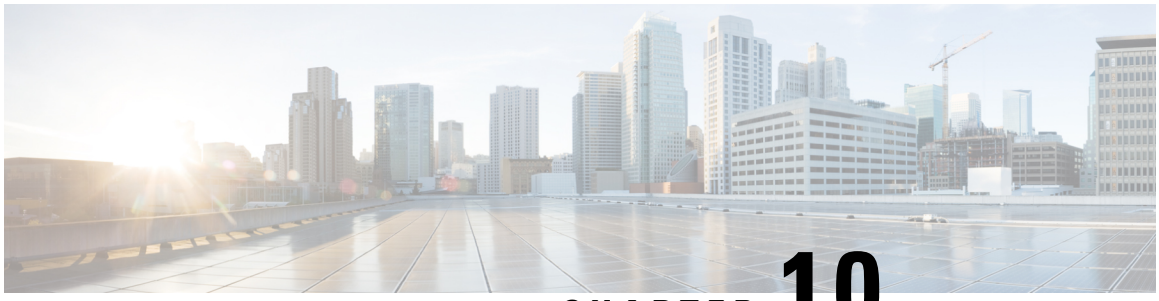
Stops the chat session.

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.

URL:	https://<ServerIP>:<Port>/ccp/chat
HTTPS method:	DELETE
HTTPS response headers:	See HTTPS Responses.

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.



CHAPTER 10

Chat Feedback

This section describes the supported commands for the chat rating.

- Post, on page 51
- Chat Feedback Parameters, on page 52

Post

This API submits the rating given by the customer after a chat conversation with the agent. This API should be called before **ChatInactivityTimeout**. API calls after this timeout will fail.

Note **ChatInactivityTimeout** refers to 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.

URL:	https://<SM_FQDN>/ccp/chat/feedback
HTTP method:	POST
Parameters:	See Chat Feedback Parameters, on page 52
Input format:	XML, JSON

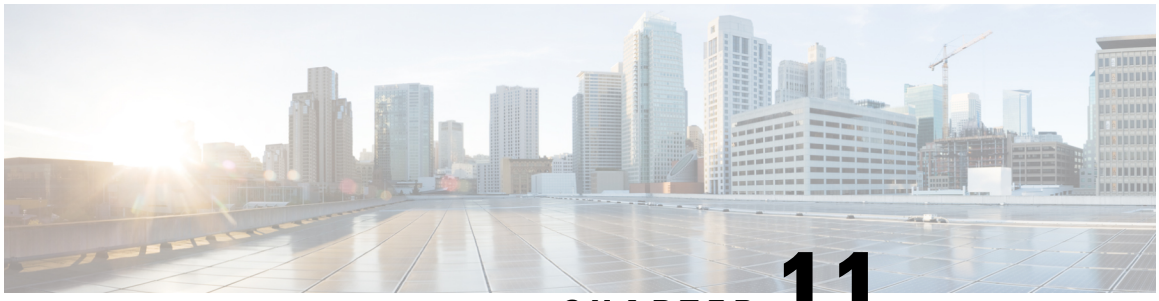
Example XML request payload:	`<Feedback>`
	`  <rating>4</rating>`
	`</Feedback>`

Example JSON request payload:	`{`
	`"rating" : "4"`
	`}`

HTTPS response codes:	200 OK 400 Bad Input 404 Not Found 405 Method Not Allowed 500 Internal Server Error See HTTPS Responses, on page 4 for more information about the response codes.

Chat Feedback Parameters

Parameter	Description	Notes
rating	Rating given by the customer.	Required The valid range is 1-5



CHAPTER 11

CCE Multichannel Routing

- Configuration for Multichannel Routing, on page 53
- Callback, on page 54
- Task, on page 58
- Variables Used for Callback and Task APIs, on page 64
- Contact States and Reason Codes, on page 68
- XMPP Eventing Versus Task API Polling, on page 70

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

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

Related Topics

GET, on page 53

PUT, on page 54

GET

Get the MR PG configuration.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/contactcenter/ mrconfig/default
HTTPS method:	GET
Example XML response:	<MRconfig> <enabled>true</enabled> <port>38001</port> <hostA>myhostname.abc.com</hostA> <hostB>ccx.host.xyz.com</hostB>

	<refURL> https://socialminer.server.ip/ccp-webapp/ ccp/contactcenter mrconfig/default </refURL> </MRconfig> See also API Conventions, on page 1.

PUT

Edit the MR PG connection configuration.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/contactcenter/mrconfig/default
HTTPS method:	PUT
Input and output format:	XML
Example XML request payload:	<MRconfig> <enabled>false</enabled> <port>38001</port> <hostA>myhostname.abc.com</hostA> <hostB>ccx.host.xyz.com</hostB> </MRconfig>

Parameters:	• **enabled**: a Boolean flag to enable or disable the MR configuration.
	  • If false: MR PG configuration is ignored and no connections are accepted.
	  • If true and no hostnames are set: any connection is accepted.
	  • If true and at least one hostname is specified: only connections matching the specified hosts are allowed.
	• **hostA, hostB**: 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.
	• **port**: the port number that the MR PG uses to connect to SocialMiner. Defaults to 38001. The valid range is 10000 - 65535.

Callback

The Callback API allows callback applications to send a notification to Unified Contact Center Enterprise (Unified 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 the estimated wait time) and to cancel previous callback requests (see DELETE, on page 57).

Note You must use the callback API to **submit or cancel** callback requests. You cannot use the social contact API.

Before you create a callback request, you must have a Callback feed (type 10) assigned to a campaign (see Feed) 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 applications can use to retrieve the status of the callback request, including the estimated wait time.

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

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

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 56.

URL:	https://<ServerIP>:<Port>/ccp/callback/feed/<callbackFeedId> The <callbackFeedId> specifies the callback feed to target for the callback request.
HTTPS method:	POST
Example XML request payload:	<Contact> <name>name</name> <title>title</title> <description>description</description> <mediaAddress>phoneNumber</mediaAddress> <tags> <tag>tag1</tag> <tag>tag2</tag> </tags> <variables> <variable> <name>cv_[1-10]</name> <value>callVariableValue</value> </variable> <variable> <name>user_(eccVariableName)</name> <value>eccVariableValue</value> </variable> <variable> <name>anythingElseExtensionFieldName</name> <value>anythingElseExtensionFieldValue</value> </variable> </variables> </Contact>

	Note The contact name, title, and mediaAddress (the phone number to be called) are required to create the request.
A reference URL to the contact is returned in the location field in the header:	https://<ServerIP>:<Port>/ccp/callback/contact/6EEF968810000132000015F60A568DFB
Response codes:	201 Created 400 Bad Request See HTTPS Responses for more information about the response codes.

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.

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

GET

Returns a reference URL for a single callback contact.

When you send a poll request, the “estimatedWaitTime” is returned only if the contact state is “Queued”.

URL:	https://<ServerIP>:<Port>/ccp/callback/contact/<ContactID>
HTTPS method:	GET
Parameters:	See Callback API Parameters.
A reference URL to the contact is returned in the location field in the header:	https://<ServerIP>:<Port>/ccp/callback/contact/6EEF968810000132000015F60A568DFB **Note** 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.

Response codes:	200 OK 404 Not Found For more information about the response codes, see HTTPS Responses, on page 4.

DELETE

Cancels a callback request. You can cancel a callback request only when the contact state is “Unread” or “Queued”.

After 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. You can use the Social Contact GET API to poll the status of the social contact. See GET, on page 174.

Note The customer will receive an error if the cancellation request is sent after an agent is already selected.

URL:	https://<ServerIP>:<Port>/ccp/callback/contact/<Id>
HTTPS method:	DELETE
Example request XML payload:	None
Response codes:	200 OK 400 Bad Request 404 Not Found See HTTPS Responses for more information about the response codes.

Callback API Parameters

Parameters are optional unless otherwise noted.

Parameter	Description	Notes
title	The name of the callback request.	Required for POST and GET (create).
name	The person who generated the callback request.	Required for POST and GET (create). The name is limited to a maximum of 100 characters. A newline (\n) is not allowed in the name.
mediaAddress	The phone number to call.	Required for POST and GET (create).

Parameter	Description	Notes
description	An optional description to accompany the callback request.	
tags	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.
variables	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_'.
status	The status of a callback request.	See "Contact State" in GET, on page 56.
estimatedWaitTime	An estimate of the amount of time (in seconds) until an agent's phone will be available to place the call.	If CCE does not have enough data to calculate estimatedWaitTime, the API returns a value of -1.

Task

The Task API allows applications to submit multichannel customer requests to Unified CCE to be routed to contact center agents. The API works in conjunction with a task feed, campaigns, and a Connection to Unified CCE Notification. This API is XML-based and not authenticated. If SocialMiner is unavailable due to an internal error and recovers, it resubmits the tasks that are in unread or queued state to Unified CCE without user intervention.

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

Note Use the Task API to submit or cancel task requests. You cannot use the social contact API.

Before you create a task request, assign a task request feed (type 12) to a campaign (see Feed) and have a Connection to CCE notification setup. The notification is triggered by a specific tag that is automatically created in the task. See Notification Rule, on page 115.

The response of the create task contains a URL in the location field that applications can use to retrieve the status of the task request, including the estimated wait time. The URL is available until the request is cancelled or until no polling is detected for at least five minutes.

Note The estimated wait time is calculated only once by Unified CCE and this value is returned on subsequent polls on SocialMiner.

Task Submission API Commands

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

POST

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

Alternatively, GET can be used to create a request. GET uses UTF-8 encoded URL parameters to provide the parameters required for the request. See GET (Create Task Request), on page 60.

URL:	https://<server>:<port>/ccp/task/feed/<TaskFeedId>
HTTPS method:	POST
Parameters:	See Task Submission API Parameters, on page 61.
Example XML request payload:	<Task> <name>name</name> <title>title</title> <description>description</description> <scriptSelector>StringIdentifier</scriptSelector> <customerUID>string</customerUID> <requeueOnRecovery>true|false</requeueOnRecovery> <!-- Indicates if the contact will be re-queued/discarded on SM failure recovery--> <tags> <tag>tag1</tag> <tag>tag2</tag> </tags> <variables> <!-- Below two fields are optional fields. 1) include mediaType to indicate the media type attribute of POD when it is created. 2) If podRefURL is passed then POD creation will be skipped for this contact. <variable> <name>mediaType</name> <value>chat</value> </variable> <variable> <name>podRefURL</name> <value>https://context-service.rciad.ciscoccservice.com/context/ pod/v1/podId/b066c3c0-c346-11e5-b3dd-3f1450b33459</value> </variable> --> <variable> <name>cv_[1-10]</name> <value>callVariableValue</value> </variable> <variable> <name>user_(eccVariableName)</name> <value>eccVariableValue</value> </variable> <variable> <name>anythingElseExtensionFieldName</name> <value>anythingElseExtensionFieldValue</value> </variable>

	</variables> </Task>
A reference URL to the contact is returned in the location field in the header:	https://<server>:<port>/ccp/task/contact/6EEF968810000132000015F60A568DFB
Response codes:	201 Created 400 Bad Request See HTTPS Responses for more information about the response codes.

GET (Create Task Request)

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

URL:	https://<server>:<port>/ccp/task/feed/<TaskFeedId>?<queryparameters>
HTTPS method:	GET
Parameters:	See Task Submission API Parameters.
A reference URL to the contact is returned in the location field in the header:	https://<server>:<port>/ccp/task/contact/6EEF968810000132000015F60A568DFB
Response codes:	201 Created 400 Bad Request See HTTPS Responses for more information about the response codes.

GET

The POST and GET methods return a URL in the location header of a successful create response. This URL is used to poll the task request status.

When you send a poll request, the “estimatedWaitTime” is returned only if the contact state is Queued. The “taskID” and “agentID” are returned only if the contact state is Handled or Reserved.

URL:	https://<server>:<port>/ccp/task/contact/<contactId>
HTTPS method:	GET
Parameters:	See Task Submission API Parameters.
Example XML response:	<Task> <estimatedWaitTime>No_Of_Seconds</estimatedWaitTime> <status>QUEUED</status> <statusReason>STATUS_REASON</statusReason> <taskID>12345_23678_2</taskID>

	<agentID>12345<agentID> </Task> See Contact States and Reason Codes, on page 68 for contact state and state reason codes.
A reference URL to the contact is returned in the location field in the header:	https://<server>:<port>/ccp/task/contact/6EEF968810000132000015F60A568DFB **Note** The URL returned in the GET or POST create request is available for polling until the request is cancelled or until there has been no polling for at least five minutes.
Response codes:	200 OK 404 Not Found For more information about the response codes, see HTTPS Responses, on page 4.

DELETE

Cancels a task request. You can cancel a task request only when the contact state is “Unread” or “Queued”.

After 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. You can use the Social Contact GET API to poll the status of the social contact. See GET, on page 174.

Note The customer will receive an error if the cancellation request is sent after an agent is already selected.

URL:	https://<server>:<port>/ccp/task/contact/<id>
HTTPS method:	DELETE
Example request XML payload:	None
Response codes:	200 OK 400 Bad Request 404 Not Found See HTTPS Responses for more information about the response codes.

Task Submission API Parameters

Parameters are optional unless otherwise noted.

Parameter	Description	Notes
title	The name of the task request.	Required for POST and GET (create).

Parameter	Description	Notes
name	The person who generated the task request.	Required for POST and GET (create). The name is limited to a maximum of 100 characters. A newline (\n) is not allowed in name.
description	An optional description to accompany the task request.	
scriptSelector	The dialed number or script selector that is used to route the request to an agent.	Obtain the script selector from the CCE configuration. The string identifier is limited to a maximum of 32 characters.
customerUID	An optional unique identifier to search for the customer before creating a POD in deployments integrated with Context Service.	
requeueOnRecovery	Indicates if the task will be requeued or discarded when SocialMiner recovers after a failure event.	Boolean. Defaults to false.
tags	One or more tags or keywords associated with the task request. **Note** A maximum of 20 tags are allowed on a social contact.	When using GET to create a task contact, use the parameter "tags" in the query string.
variables	A set of custom name and value pairs.	When using GET to create a task request, pass the variables as query parameters and start with "variable_". You can specify the pre-defined variables "mediaType" and "podRefURL" for a task request. These are used for Context Service. For more information, see Variables Used for Callback and Task APIs, on page 64.
status	The status of a task request.	See Contact States and Reason Codes, on page 68.
estimatedWaitTime	An estimate of the amount of time (in seconds) until an agent is available to handle the task.	Returned in the XML success response. If CCE does not have enough data to calculate estimatedWaitTime, the API returns a value of -1.

Parameter	Description	Notes
taskId	The unique ID that is assigned by Unified CCE for a task request.	Returned in the XML success response.
agentId	The ID of the agent who is assigned by Unified CCE for a task request.	Returned in the XML success response.

Task Resubmission API Commands

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

PUT

Enables resubmission of an existing task request with modified attributes to Unified CCE.

URL:	https://<SM FQDN>/ccp-webapp/ccp/task/{contact id}/renotify
HTTPS method:	PUT
Input/Output format:	Application, XML
Parameters:	See Task Resubmission API Parameters, on page 63.

Example XML request payload:	`<Task>`
	`<scriptSelector>new_or_same_script_selector`
	`</scriptSelector>`
	`<reNotifyReason>ROUTE_SERVICE_RECOVERY`
	`</reNotifyReason>`
	`</Task>`

HTTPS response headers:	200 OK 400 Bad Request 404 Not Found 412 Precondition Failed 500 Internal Server Error

Task Resubmission API Parameters

Parameters are optional unless otherwise noted.

Parameter	Description	Notes
scriptSelector	The dialed number or script selector that is used to route the request to an agent.	This is a mandatory field. Obtain the script selector from the CCE configuration. The string identifier is limited to a maximum of 32 characters.

Parameter	Description	Notes
reNotifyReason	Indicates the reason for renotification. • ROUTE_SERVICE_TRANSFER—Include for transfer. • ROUTE_SERVICE_RECOVERY—Include for failure recovery.	

Task Closure API Commands

This section describes the PUT command for the Task Closure API.

PUT

Closes an existing task request.

URL:	https://<SM FQDN>/ccp-webapp/ccp/task/{contact id}/close
HTTPS method:	PUT
HTTPS method:	Input—None Output—Application, XML
Request Headers	Authorization: Basic
HTTPS response headers:	200 OK 400 Bad Request 401 Unauthorized 404 Not Found 412 Precondition Failed 500 Internal Server Error

Variables Used for Callback and Task APIs

Use the following variables to pass values as part of the callback or task API requests.

Call Variables

When SocialMiner initiates a callback or task request with Unified CCE then the variables cv_1 to cv_10 are included as call variables. The prefix “cv_” is not case-sensitive.

ECC Variables

When SocialMiner initiates a callback or task request with Unified CCE then the variables that start with the string “user_” are included as Expanded Call Context (ECC) variables. The prefix “user_” is not case-sensitive.

If you send an ECC variable that is not configured in Unified CCE, then it does not result in failure. Unified CCE ignores the variable.

Extension Fields

All the variables that are associated with the request in the XML are included as extension fields in the SocialMiner callback or task contact.

To specify the media type of POD for Context Service, pass a variable with the name “mediaType” and set a value from the following list. If you don't specify the variable mediaType, then the media type “event” is set to the Piece Of Data (POD).

- chat
- email
- event
- mobile
- social
- video
- voice
- web

To pass the POD reference URL that is already created for the incoming task request to Unified CCE, pass a variable with the name “podRefURL”, with URL as the value. If you don't specify the variable podRefURL, then SocialMiner creates the POD.

Examples

The following are the examples to create a SocialMiner callback or task contact using POST and GET.

Note These examples do not include all the SocialMiner callback or task contact attributes.

Create SocialMiner Callback / Task Contact using POST

The following example shows how to submit a new request with call variable “cv_7” set to “test7”, ECC variable “user_user.callback.test” set to “ct7”, extension field location set to “Boston”, and with “tags”.

For callback:

```
<Contact>
   <name>Customer</name>
   <title>Help</title>
   <mediaAddress>5551212</mediaAddress>
    <tags>
      <tag>tag1</tag>
      <tag>tag2</tag>
      <tag>tag2</tag>
    </tags>
   <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>
```

For task:

```
<Task>
   <name>Customer</name>
   <title>Help</title>
   <scriptSelector>5551212</scriptSelector>
   <requeueOnRecovery>true
   </requeueOnRecovery>
   <tags>
      <tag>tag1</tag>
      <tag>tag2</tag>
      <tag>tag2</tag>
    </tags>
   <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>
      <variable>
         <name>mediaType</name>
         <value>chat</value>
      </variable>
      <variable>
         <name>podRefURL</name>
         <value>https://context-
         service.rciad.ciscoccservice.com/
         context/pod/v1/podId/b066c3c0-c346
         -11e5-b3dd-3f1450b33459</value>
      </variable>
   </variables>
</Task>
```

Create SocialMiner Callback / Task Contact using GET

The following example shows how to submit a new request with call variable “cv_7” set to “test7”, ECC variable “user_user.callback.test” set to “ct7”, extension field location set to “Boston”, and with “tags” parameter in the query string.

When you use GET to create a request, variables must be passed as query parameters. Variables are denoted by the “variable_” prefix.

For callback:

```
https://sample_server/ccp/callback/feed/12345?name=Customer&title=Help&mediaAddress=5551212
&variable_cv_7=test7&variable_user_user.callback.test=ct7&variable_location=Boston&tags=tag1,tag2,tag3
```

For task:

```
https://sample_server/ccp/task/feed/100093?name=Customer&title=Help&description=
some_desc&scriptSelector=5551212&requeueOnRecovery=true&variable_podRefURL=https://context
-service-uq/context/v1/skjdskdgskd&variable_mediaType=chat&variable_cv_7=test7
&variable_user_user.callback.test=ct7&variable_location=Boston&tags=tag1,tag2,tag3
```

Creation of SocialMiner Social Contact

The above examples for POST and GET results in the creation of the following SocialMiner social contact:

For callback:

```
<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>
   <tags>
      <tag>tag1</tag>
      <tag>tag2</tag>
      <tag>tag3</tag>
   </tags>
</SocialContact>
```

For task:

```
<SocialContact>
   <author>Customer</author>
   <title>Help</title>
   <description>some_desc</description>
      <extensionFields>
      <extensionField>
         <name>scriptSelector</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>
      <extensionField>
         <name>mediaType</name>
         <value>chat</value>
      </extensionField>
      <extensionField>
         <name>podRefURL</name>
         <value>https://context-
         service.rciad.ciscoccservice.com/
         context/pod/v1/podId/b066c3c0-c346
         -11e5-b3dd-3f1450b33459</value>
      </extensionField>
   </extensionFields>
   <status>handled</status>
   <statusReason>externally_handled
   </statusReason>
   <tags>
      <tag>tag1</tag>
      <tag>tag2</tag>
      <tag>tag3</tag>
      <tag>requeueOnRecovery</tag>
   </tags>
</SocialContact>
```

Contact States and Reason Codes

The application developer is responsible for the messages that their application's interface provides to the customer. The application developer can use the <status> and <statusReason> fields to understand and write an appropriate message for the application interface to display to the customer for a state.

Contact States and Reason Codes for Callback Request State Transitions

The following table lists the contact state transitions as the callback request is processed.

Contact State	State Reason Code	Description
Queued	EXTERNALLY_HANDLED	The request is successfully submitted to the contact center.
Handled	EXTERNALLY_HANDLED	The request is successfully routed to an agent.

Contact States and Reason Codes for Task Request State Transitions

The following table lists the contact state transitions as the task request is processed.

Contact State	State Reason Code	Description
Queued	TASK_RENOTIFY_REQUESTED	The request resubmission to the contact center is under process.
	EXTERNALLY_ROUTED	The request is successfully submitted to the contact center.
Reserved	EXTERNALLY_ROUTED	The request is successfully routed to an agent.
Handled	EXTERNALLY_HANDLED	The request is successfully completed by an agent.

Reason Codes for Callback and Task Request Discarded State Transition

The following table lists the discarded contact state transition. Check the Unified CCE configuration to rectify.

Note The application must resubmit the discarded requests; they are not automatically resubmitted.

Contact State	State Reason Code	Description
Discarded	NOTIFICATION_INVALID_NEW_TASK_MESSAGE	The message in the route request is invalid.
	NOTIFICATION_MEDIA_ROUTING_DISABLED	The routing is disabled.
	NOTIFICATION_NO_SCRIPT	The routing script does not exist.
	NOTIFICATION_INVALID_MRD_ID Only applicable for callback.	The Media Routing Domain (MRD) ID does not exist. Ensure the MRD ID that is configured in the SocialMiner notification exists in the Unified CCE configuration.
	NOTIFICATION_INVALID_SCRIPT_SELECTOR	The dialed number or script selector does not exist. Ensure the dialed number or script selector that is configured in the SocialMiner notification exists in the Unified CCE configuration.
	NOTIFICATION_ROUTER_RELEASED_TASK	The task is released by Unified CCE as the script indicates that the request be dropped. Possible script misconfiguration.
	NOTIFICATION_UNKNOWN_ROUTING_PROBLEM	Unknown routing problem.
	NOTIFICATION_CCE_CONNECTION_LOST	The connection to Unified CCE is either lost or is not established.
	NOTIFICATION_CCE_SOCIALMINER_SYSTEM_FAILURE	While the callback is queued, SocialMiner either failed or restarted.
	NOTIFICATION_INVALID_VARIABLE	SocialMiner could not submit the task to Unified CCE due to invalid values in the notification. The following are the maximum field lengths that are allowed: • media address: 39 bytes (Only applicable for callback) • call variable: 40 bytes • ECC variable name: 32 bytes • ECC variable value: 210 bytes
	NOTIFICATION_RATE_LIMITED	The incoming rate of requests exceeded the allowed number of contacts. The following are the maximum contacts that are allowed: • Callback API: 40 contacts per minute • Task API: 8 contacts per second

Contact States and Reason Codes for Callback Cancel Requests

The following table lists the contact states and reason codes that the cancellation requests trigger for callback contacts.

Social Contact State	Status Reason Code	Description
Queued	NOTIFICATION_CCE_CALLBACK_ CANCEL_REQUESTED	Cancel callback request was initiated.
Discarded	NOTIFICATION_CCE_CALLBACK_ CANCEL_SUCCEEDED	Unified CCE cancelled the callback request successfully.
Handled	EXTERNALLY_HANDLED	The cancellation request failed because the task was routed to an agent before the callback request could be cancelled.

Contact States and Reason Codes for Task Cancel Requests

The following table lists the contact states and reason codes that the cancellation requests trigger for tasks.

Social Contact State	Status Reason Code	Description
Queued	NOTIFICATION_CCE_TASK_ CANCEL_REQUESTED	Cancel task request was initiated.
Discarded	NOTIFICATION_CCE_TASK_ CANCEL_SUCCEEDED	Unified CCE cancelled the task request successfully.
Reserved	EXTERNALLY_ROUTED	The cancellation request failed because the task was routed to an agent before the task request could be cancelled.

XMPP Eventing Versus Task API Polling

For applications that create many tasks at once, such as email, using the Task API to poll for status can be inefficient. Instead, you can use XMPP eventing to monitor Task campaign events.

Status Polling with the Task API

When you configure SocialMiner for Universal Queue capability, the system creates a Task feed automatically. That feed is associated with a Task campaign. All tasks created using the Task API result in contacts associated with this campaign. The contacts IDs are the IDs at the end of the reference URLs returned when a task is submitted.

You use these reference URLs to poll for task status.

For example, this task URL:

```
https://[ServerIP]:[Port]/ccp/task/contact/3674633110000152000014D3F57FECE
```

maps to this contact:

```
https://[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/3674633110000152000014D3F57FECE
```

Subscribe to XMPP Task Campaign Events

When the system creates the Task campaign, it also creates the ccp.campaign.updates.<*campaignpublicId*> node that allows subscribers to subscribe to XMPP events related to the campaign.

Note This is secured using TLS 1.2 by default, but the administrator can modify its security parameters. For more information, see Cisco SocialMiner User Guide <Insert Link here>

To subscribe to the node for the Task campaign, you need the campaign's public ID. First, find the Task feed ID, using the Feed GET (List) API. Then, you can find the campaign's public ID using the Campaign GET (List) API.

In this example Feed GET (List) response, the application is using the feed ID 100000.

```
https://[ServerIP]Port]/ccp-webapp/ccp/feed

<feeds>
  ...
  <Feed>
    <changeStamp>0</changeStamp>
    <description>Feed for Universal Queuing</description>
    <name>Cisco_Default_Task_Feed</name>
    <pushFeedURL>https://[ServerIP]:[Port]/ccp/task/feed/100000</pushFeedURL>
    <refURL>https://[ServerIP]:[Port]/ccp-webapp/ccp/feed/100000</refURL>
    <status>1</status>
    <tags>
      <tag>cisco_uq_tag</tag>
    </tags>
    <type>12</type>
  </Feed>
  ...
</feeds>
```

In this example Campaign GET (List) response, the publicID associated with that feed is Cisco_Default_Task_Campaign. The application therefore subscribes to **ccp.campaign.updates.Cisco_Default_Task_Campaign**.

```
<campaigns>
  ...
  <Campaign>
    <changeStamp>0</changeStamp>
    <description>Campaign for any type of task</description>
    <feeds>
      <feed>https://[ServerIP]Port]/ccp-webapp/ccp/feed/100000</feed>
    </feeds>
    <name>Cisco_Default_Task_Campaign</name>
    <publicId>Cisco_Default_Task_Campaign</publicId>
    <refURL>https://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/
     Cisco_Default_Task_Campaign</refURL>
    <resultsURL>https://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/
     Cisco_Default_Task_Campaign/results</resultsURL>
    <suggestedTagsURL>https://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/
     Cisco_Default_Task_Campaign/suggestedtags</suggestedTagsURL>
  </Campaign>
  ...
</campaigns>
```

For more information, see:

- Feed API GET (List), on page 91

- Campaign API GET (List), on page 20

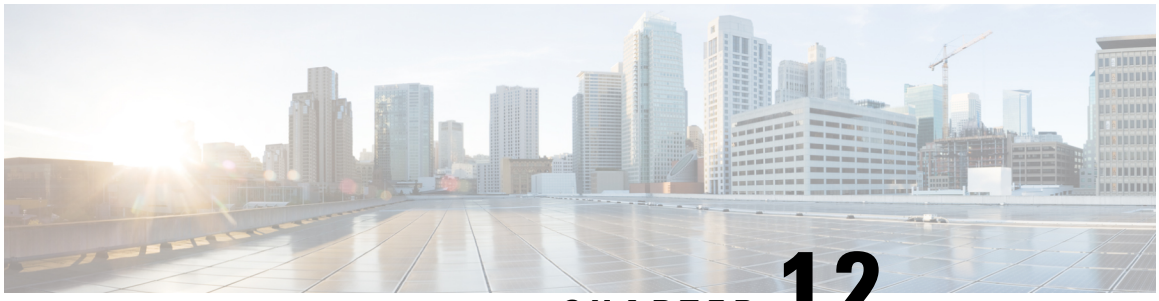
Mapping Social Contact Extension Parameters and Task API Parameters

When you use XMPP eventing, your application receives social contact data that provides the content that backs the Task API. The following table maps social contact extension fields to Task API fields.

Social Contact Extension Field	Task API Field
scriptSelector	scriptSelector
cv_[1-10]	variables.variable.cv_[1-10]
user_ (eccVariableName)	variables.variable.user_(eccVariableName)

Sample Task Campaign XMPP Events

```
<SocialContact xmlns="http://jabber.org/protocol/pubsub">
  <campaignPublicId>Cisco_Default_Task_Campaign</campaignPublicId>
  <extensionFields>
    <extensionField><name>scriptSelector</name><value>smChatIntScriptSelector</value>
    </extensionField>
  </extensionFields>
  <id>3E5C39DE10000152000002B43F57FECE</id>
  <publishedDate>1452745177552</publishedDate>
  <refURL>https://[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/
   3E5C39DE10000152000002B43F57FECE
  </refURL>
  <sourceType>task</sourceType>
  <status>unread</status>
  <statusReason>unknown</statusReason>
  <statusTimestamp>1452745177566</statusTimestamp>
  <statusUserId/>
  <tags><tag>task</tag></tags>
</SocialContact>
```



CHAPTER 12

Context Service

The Context Service API allows you to update the context service configuration in SocialMiner including the registration status and the campaigns that are associated with the context service script filter.

For more information about Context Service, see the "Context Service" section of *Cisco Unified Contact Center Express Design Guide*, located at:

https://www.cisco.com/c/en/us/support/customer-collaboration/unified-contact-center-express/products-implementation-design-guides-list.html.

- Context Service API Commands, on page 73

Context Service 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 73
PUT, on page 74
Context Service Configuration API Parameters, on page 75

GET

Retrieves the context service configuration.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/contextServiceConfig
HTTPS method:	GET
Parameters:	See Context Service Configuration API Parameters, on page 75.

Example XML request payload:	`<ContextServiceConfig>`
	`<connectionData>********</connectionData>`
	`<status>REGISTERED</status>`
	`<requestTimeoutInMS>1900</requestTimeoutInMS>`
	`<maxRetries>5</maxRetries>`
	`<labmode>true</labmode>`
	`<proxyurl>https://[ServerIP]:[Port]</proxyurl>`
	`<campaigns>`

	<campaign>https://[ServerIP]/ccp-webapp/ccp/campaign/CCX_Chat_Campaign</campaign> <campaign>https://[ServerIP]/ccp-webapp/ccp/campaign/CCX_Email_Campaign</campaign> </campaigns> </ContextServiceConfig>
HTTPS response headers:	401 Unauthorized 500 Internal Server Error See HTTPS Responses, on page 4.

PUT

Updates the context service configuration.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/contextServiceConfig You can access this URL only through https.
HTTPS method:	PUT
Parameters:	See Context Service Configuration API Parameters, on page 75.
Example XML request payload:	<ContextServiceConfig> <connectionData>abcd</connectionData> <requestTimeoutInMS>1900</requestTimeoutInMS> <maxRetries>5</maxRetries> <labmode>true</labmode> <proxyurl>[ServerIP]:[Port]/proxyurl> <status>REGISTERED</status> <campaigns> <campaign>https://[ServerIP]/ccp-webapp/ccp/campaign/CCX_Chat_Campaign</campaign> <campaign>https://[ServerIP]/ccp-webapp/ccp/campaign/CCX_Email_Campaign</campaign> </campaigns> </ContextServiceConfig>
HTTPS response headers:	400 Bad Request 401 Unauthorized 403 Forbidden 500 Internal Server Error See HTTPS Responses, on page 4.

Re-Initialize Context Service

PUT

Re-initializes the context service client in SocialMiner.

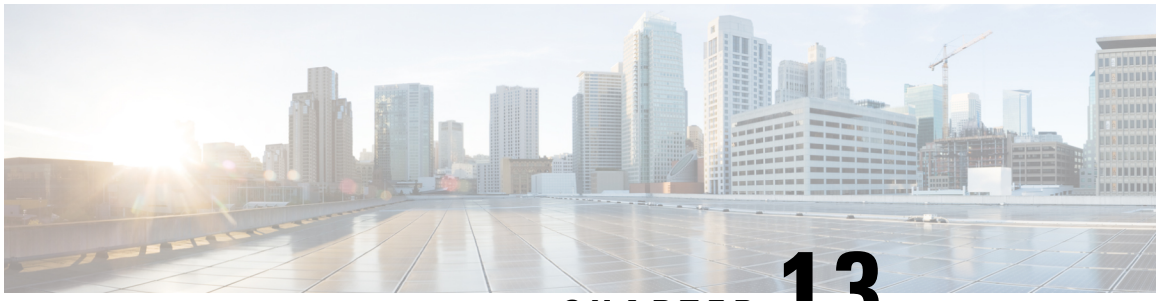
URL:	https://<SM FQDN>/ccp-webapp/ccp/contextServiceConfig/init

	You can access this URL only through https.
HTTPS method:	PUT
Parameters:	None
HTTPS response headers:	401 Unauthorized 403 Forbidden 500 Internal Server Error See HTTPS Responses, on page 4.

Context Service Configuration API Parameters

Parameters are optional unless otherwise noted.

Parameter	Description	Notes
connectionData	Used for registering SocialMiner with Context Service Cloud.	String. Maximum of 4096 bytes (UTF-8). If the status is REGISTERED, then <connectionData> value cannot be empty or null.
status	Status of SocialMiner registration.	String. Values are: • REGISTERED • UNREGISTERED
requestTimeoutInMS	The time in milliseconds for the connection request to be timed out.	
maxRetries	Maximum number of retries allowed for any Context Service operation.	
labmode	Context Service lab mode setting.	String. Values are: • True • False
proxyurl	Proxy url to be used by Context Service client to communicate with the Context Service cloud.	String.
campaign	Valid URL of the campaign.	



CHAPTER 13

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, on page 77

Email API Commands

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

Related Topics

GET, on page 77
PUT, on page 78
Email API Parameters, on page 78

GET

Retrieves the SMTP configuration.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/email/default
HTTPS method:	GET
Example XML response:	<Email> <smtpHost><ServerIP>:<Port></smtpHost> <smtpPort>587</smtpPort> <smtpFromUser> FromUser@Here.net </smtpFromUser> <smtpHostUserName> userNameForEmailServer </smtpHostUserName> <smtpAuthenticationEnabled>

	<pre> true
	  </smtpAuthenticationEnabled>
	  <smtpEnabled>true</smtpEnabled>
	  <smtpSslEnabled>true</smtpSslEnabled>
	  <refURL>
	  https://<ServerIP>:<Port>/ccp-webapp/ccp/
	  email/default
	</refURL>
	</Email></pre>

Parameters:	See Email API Parameters, on page 78.

PUT

Updates the SMTP configuration.

Note Only the administrator created during install can use this API.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/email/default
HTTPS method:	PUT

Example XML request payload:	<pre><Email>
	  <smtpHost>[ServerIP]</smtpHost>
	  <smtpPort>587</smtpPort>
	  <smtpFromUser>FromUser@Here.net</smtpFromUser>
	  <smtpHostUserName>userNameForEmailServer</smtpHostUserName>
	
	<smtpHostUserPassword>userPasswordForEmailServer</smtpHostUserPassword>
	
	  <smtpAuthenticationEnabled>true</smtpAuthenticationEnabled>
	  <smtpEnabled>true</smtpEnabled>
	  <refURL>https://[ServerIP]/ccp-webapp/ccp/email/default</refURL>
	</Email></pre>

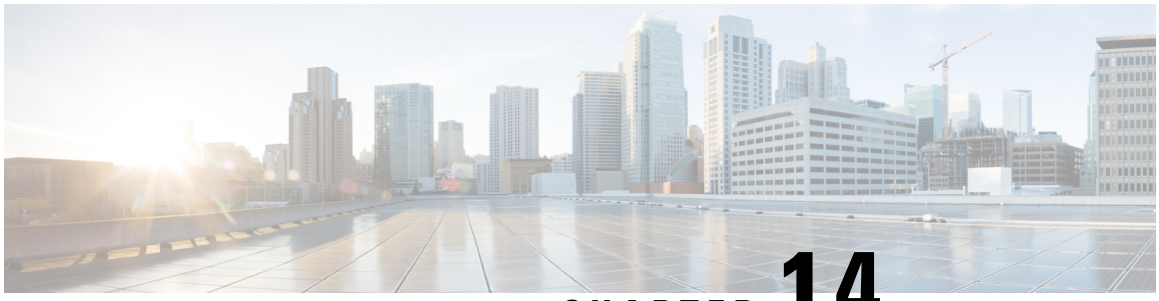
Parameters:	See Email API Parameters, on page 78.

Email API Parameters

Parameters are optional unless otherwise noted.

Parameter	Description	Notes
smtpEnabled	Whether this SMTP configuration is enabled.	Boolean. Defaults to false.
smtpAuthenticationEnabled	Whether SMTP authentication is required for the SMTP host.	Boolean. Defaults to false.
smtpFromUser	The email (reply-to) address of email sent by this server.	Required when smtpEnabled = true.

Parameter	Description	Notes
smtpHost	The fully qualified host address of the SMTP server.	Required when smtpEnabled = true.
smtpHostUserName	The username used to log into the SMTP server.	Required when smtpAuthenticationEnabled = true.
smtpHostUserPassword	The password used to log into the SMTP server.	Required when smtpAuthenticationEnabled = true.
smtpPort	The SMTP port. Default is 587.	Required when smtpEnabled = true.
smtpSslEnabled	Whether SSL is enabled for the SMTP server.	Boolean. Defaults to true.



CHAPTER 14

Email Reply

The Email Reply API allows you to respond to email contacts. You must configure an email feed before you can use this API.

- Email Reply API Commands, on page 81

Email Reply API Commands

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

GET (Email)

Instantiates a server task to get the body of an email contact and any customer-side attachments from the email server. The client polls the URL provided in the location header to get the status of the task.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/reply/email/<id>
HTTPS method:	GET
Input/Output format:	xml
Parameters:	See Email Reply API Parameters, on page 85.
Example XML response:	<EmailMessage> <id>072D0E871000012B0000ED8B0A568DDF</id> <subject>Email subject</subject> <fromAddress>sender@xyz.com</fromAddress> <toAddress>name1@example.com</toAddress> <ccAddress>name2@example.com,name3@example.com</ccAddress> <receivedTimestamp>1302551491320</receivedTimestamp> <contentType>text/html</contentType> <body><![CDATA[...]]></body> <refDraftURL>https://<server>:<serverport>/ccp-webapp/ccp/reply/email/{id}/draft</refDraftURL> <attachments> <incomingAttachments> <attachment> <name>incoming_filename</name> <refURL>https://<server>:<serverport>/ccp-webapp/ccp/reply/email/<id>/attachment/incoming/ <attachmentIndex></refURL>

	<sizeBytes>256</sizeBytes> </attachment> ... </incomingAttachments> </attachments> </EmailMessage>
HTTPS response headers:	HTTPS response: • 200—Task is finished with result. • 202—Task is in progress. Poll again. Location header: https://<Server>:<Port>/ccp-webapp/ccp/reply/email/F99A8E33100001470000004A0A56863B/getemailbody/1234

GET (Email Reply Draft Data)

Instantiates a server task to retrieve email reply draft data, including the email reply draft content and any agent-side attachments. This information is available while the agent is working on the email reply draft or when the email reply draft is saved and requeued . After the email reply is sent, the information is not available.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/reply/email/<id>/draft
HTTPS method:	GET
Input/Output format:	xml
Parameters:	See Email Reply API Parameters, on page 85.
Example XML response:	<EmailMessage> <id>072D0E871000012B0000ED8B0A568DDF</id> <subject>Email subject</subject> <fromAddress>sender@xyz.com</fromAddress> <toAddress>name1@example.com</toAddress> <ccAddress>name2@example.com,name3@example.com</ccAddress> <bccAddress>name4@example.com, name5@example.com</bccAddress> <ReplyToAddress>name6@example.com</ReplyToAddress> <receivedTimestamp>1302551491320</receivedTimestamp> <contentType>text/html</contentType> <body><![CDATA[...]]></body> <attachments> <outgoingAttachments> <attachment> <name>filename</name> <refURL>https://<server>:<serverport>/ccp-webapp/ccp/reply/email/<id>/attachment/outgoing/ <fileIdentifier></refURL> <sizeBytes>100</sizeBytes> </attachment> ... </outgoingAttachments> </attachments> </EmailMessage>
HTTPS response headers:	HTTPS response: 202 Location header:

	`https://<Server>:<Port>/ccp-webapp/ccp/reply/email/F99A8E33100001470000004A0A56863B/getreplydraft/1234` The client polls the URL provided in the location header to get the status of the task. URL: Provided in the location header HTTPS method: GET Input/Output format: xml Response: • 200—Task is finished with result. • 202—Task is in progress. Poll again.

POST (Create Email Reply)

Instantiates a server task to send an email reply to the mail server. The client polls the URL provided in the location header to get the status of the task.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/reply/email/<id>
HTTPS method:	POST
Input/Output format:	xml
Parameters:	See Email Reply API Parameters, on page 85.

Example XML request payload:	`<EmailMessage>`
	`<toAddress>sender@xyz.com,name@example.com,name1@example.com</toAddress>`
	`<ccAddress>name2@example.com,name3@example.com,name4@example.com</ccAddress>`
	`<bccAddress>name5@example.com,name6@example.com,name7@example.com</bccAddress>`
	`<ReplyToAddress>name8@example.com</ReplyToAddress>`
	`<contentType>text/html</contentType>`
	`<body><![CDATA[ ... ]]></body>`
	`</EmailMessage>`

HTTPS response headers:	HTTPS response: • 202—Task is in progress. Poll again. Location header: `https://<Server>:<Port>/ccp-webapp/ccp/reply/email/EE9A8E33100001470000004A0A5686AA/sendemail/4567`

POST (Create Email Forward)

Instantiates a server task to send an email foward to the mail server. The client polls the URL provided in the location header to get the status of the task.

URL:	https://<server>:<serverport>/ccp-webapp/ccp/reply/email/{id}?emailAction=forward
HTTPS method:	POST
Input/Output format:	xml
Parameters:	See Email Reply API Parameters, on page 85.
Example XML request payload:	<EmailMessage> <toAddress>sender@xyz.com,name@example.com,name1@example.com</toAddress> <ccAddress>name2@example.com,name3@example.com,name4@example.com</ccAddress> <bccAddress>name5@example.com,name6@example.com,name7@example.com</bccAddress> <replyToAddress>mdinkleplotz@hotmail.com</replyToAddress> <contentType>text/html</contentType> <body><![CDATA[...]]></body> </EmailMessage>
HTTPS response headers:	HTTPS response: • 202—Task is in progress. Poll again. Location header: https://<Server>:<Port>/ccp-webapp/ccp/reply/email/EE9A8E33100001470000004A0A5686AA/sendemail/4567

PUT (Email Draft)

Instantiates a server task to save the email reply body as a draft email in the Drafts folder (the original body of the email is not saved).

After the email reply is sent, the draft email is automatically deleted from the Drafts folder.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/reply/email/<id>/draft
HTTPS method:	PUT
Input/Output format:	xml
Parameters:	See Email Reply API Parameters, on page 85.
Example XML request payload:	<EmailMessage> <contentType>text/html</contentType> <body><![CDATA[...]]></body> </EmailMessage>
HTTPS response headers:	HTTPS response: 202

	Location header: `https://<Server>:<Port>/ccp-webapp/ccp/reply/email/EE9A8E33100001470000004A0A5686AA/draft/4567` The client polls the URL provided in the location header to get the status of the task. URL: Provided in the location header HTTPS method: GET Input/Output format: xml, json Response: • 200—Task is finished. Email sent successfully. • 202—Task is in progress. Poll again.

Delete (Email Draft)

Deletes all draft email messages that are associated with the specified social contact from the Drafts folder of the associated IMAP account.

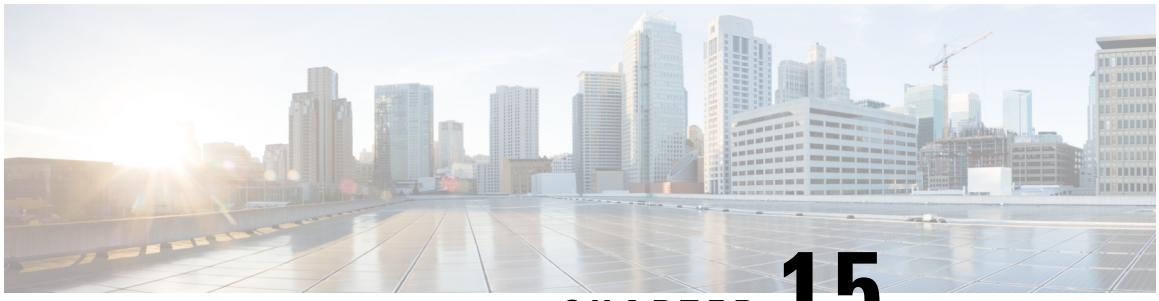
URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/reply/email/<contactid>/draft
HTTPS method:	DELETE
HTTPS response headers:	200 OK 404 Not Found For more information about the response codes, see HTTPS Responses.

Email Reply API Parameters

This table defines the parameters used by the Email Reply API.

Parameter name	Description	Notes
id	The alphanumeric ID of the social contact.	
subject	The subject line of the email.	This is also available via the social contact as "title.
fromAddress	The email address in the reply-to header (if the reply-to header is present) or the email address of the sender.	This is also available via the social contact as "author".
toAddress	The email address of the SocialMiner email feed (the address to which the email was sent).	

Parameter name	Description	Notes
ccAddress	The email address to which the email was cc'd by the sender or the agent.	
bccAddress	The email address to which the email was bcc'd by the agent.	
ReplyToAddress	The email address to which the email was replied to.	
receivedTimestamp	The timestamp (long) to indicate when the mail server received the email.	This is also available via the social contact as "published date".
contentType	The content type of the email body.	Permitted values are text/plain or text/html.
body	The main body (text) of the email enclosed within a Character Data (CDATA) section.	Required for POST (create an email reply). The other email reply parameters can be sent but only the body is used. All other information is retrieved from the social contact or from the mail server itself.
refDraftURL	The refURL to get the reply draft data (the draft email reply and agent-side attachments).	
attachments	Meta information about attachments related to the email contact.	
->outgoingAttachments	Details about attachments that agents upload in reply to a message from a customer.	
->incomingAttachments	Details about attachments that are contained in the email message from the customer.	
-->attachment	Details about one specific attachment. This parameter contains the following information: • name—the filename of the attachment • refURL—The URL to access the attachment • sizeBytes—The size of the attachment in bytes	



CHAPTER 15

IMAP Folder Configuration

• IMAP Folder Configuration, on page 87

IMAP Folder Configuration

This section outlines the API(s) available to a user to configure the IMAP folder settings which SocialMiner uses to reply to emails.

GET

Retrieves the IMAP folder configuration settings to be used for replying to emails.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/imapFolderConfig
HTTPS method:	GET
Input/Output format:	xml
Parameters:	See IMAP Folder Configuration Parameters, on page 88.

Example XML response:	`<Folders>`
	`  <Folder>`
	`  <type>drafts</type>`
	`  <displayName>[Gmail]/Drafts</displayName>`
	`  </Folder>`
	`  <Folder>`
	`  <type>sent<type>`
	`  <displayName>[Gmail]/Sent Mail</displayName>`
	`  </Folder>`
	`  <Folder>`
	`  <type>outbox<type>`
	`  <displayName>Agent-outbox</displayName>`
	`  </Folder>`
	`</Folders>`

HTTPS response headers:	HTTPS response: • 401—UNAUTHORIZED. No valid authorization key was specified. • 500—INTERNAL SERVER ERROR. An error occurred in SocialMiner server during request processing.

PUT

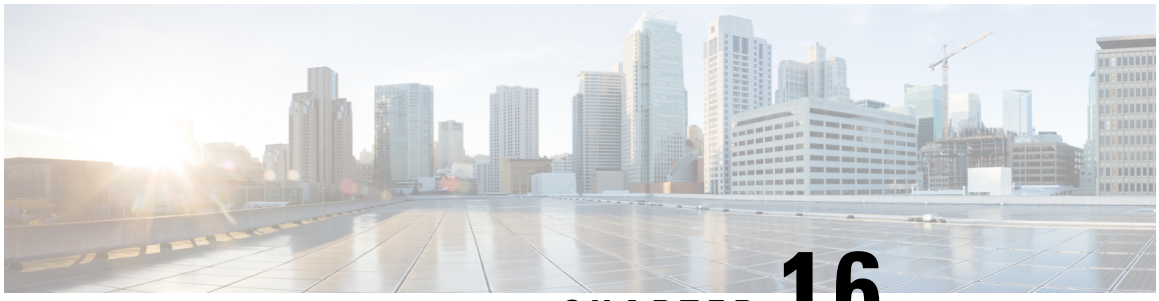
Updates the IMAP folder configuration.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/imapFolderConfig
HTTPS method:	PUT
Input/Output format:	xml
Parameters:	See IMAP Folder Configuration Parameters, on page 88.
Example XML request payload:	<Folders> <Folder> <type>drafts<type> <displayName>[Gmail]/Drafts</displayName> </Folder> <Folder> <type>outbox<type> <displayName>Agent-outbox</displayName> </Folder> <Folder> <type>sent<type> <displayName>[Gmail]/Sent Mail</displayName> </Folder> </Folders>
HTTPS response headers:	200-OK. 401- UNAUTHORIZED. 400- BAD REQUEST. 500- INTERNAL SERVER ERROR.

IMAP Folder Configuration Parameters

This table defines the parameters used by the IMAP Folder Configuration API.

Parameter name	Description	Notes
type	The type of the folder created.	Can be one of the following: • draft • sent • outbox
displayName	The name of the folder created.	This can be localized depending on the actual name of the folder in the mail server. A Maximum of 225 characters is supported for the name chosen.



CHAPTER 16

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, on page 89
- POST (Create an IMAP Email Feed), on page 90
- DELETE, on page 91
- GET (List), on page 91
- GET, on page 93
- PUT, on page 94
- Feed API Parameters, on page 94
- Email Feed API Parameters, on page 97
- Email Feed Limitations, on page 99

POST

Creates a feed to be stored in the database.

Note If you are creating an email feed, see POST (Create an IMAP Email Feed), on page 90.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/feed
HTTPS method:	POST
Parameters:	See Feed API Parameters, on page 94.

Example XML request payload:	`<Feed>`
	`  <chatInactivityTimeout>300</chatInactivityTimeout>`
	`  <chatJoinTimeout>60</chatJoinTimeout>`
	`  <description>My web chat feed</description>`
	`  <name>My Chat Feed</name>`
	`  <pushFeedURL>https://[Server]:`
	`  [Port]/ccp-webapp/ccp/chatfeed/100159</pushFeedURL>`
	
	`<refURL>https://[Server]:[Port]/ccp-webapp/ccp/feed/100159</refURL>`
	
	`  <replyTemplateRefURL>`

	<pre> https://[Server]:[Port]/ccp-webapp/ccp/template/reply/302 </replyTemplateRefURL> <status>1</status> <tags> <tag>tag1</tag> <tag>tag2</tag> <tag>tag3</tag> </tags> <type>8</type> </Feed></pre>

HTTPS response headers:	The response contains the **URL** for the newly created feed. Note the id: *100162*.
	
	<pre>https/1.1 201 Created
	Location: https://<ServerIP>:<Port>
	 /ccp-webapp/ccp/feed/100162
	Content-Type: text/plain
	Content-Length: 0
	Date: Tue, 12 Jan 2010 16:15:04 GMT</pre>
	See also HTTPS Responses.

POST (Create an IMAP Email Feed)

Creates an IMAP email feed to be stored in the database.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/feed
HTTPS method:	POST
Parameters:	See Email Feed API Parameters, on page 97.

Example XML request payload:	<pre><Feed>
	  <type>11</type>
	  <name>name of email feed</name>
	  <description>description of email feed</description>
	  <email>
	  <receive>
	  <host>imap.email.com</host>
	  <port>993</port>
	  <folderName>Inbox</folderName>
	  <snapshotAge>120</snapshotAge>
	  </receive>
	  <send>
	  <host>smtp.email.com</host>
	  <port>587</port>
	  </send>
	  <username>me@email.com</username>
	  <password>******</password>
	  </email>
	  <pollingInterval>60</pollingInterval>
	  <replyTemplateRefURL>https://10.1.1.1/ccp-webapp/ccp/
	  template/reply/105678</replyTemplateRefURL>
	  <tags>
	  <tag>tag1</tag>
	  <tag>tag2</tag>
	  </tags>
	</Feed></pre>

HTTPS response headers:	The Location field of the response header contains the reference URL for the newly created feed.
	
	`Status: 201 Created`
	`Location: https://<ServerIP>:<Port>/ccp-webapp/ccp/feed/128356`
	`Content-Type: text/plain`
	`Content-Length: 0`
	`Date:  Fri, 11 Jul 2014 19:33:21 GMT`
	
	See also HTTPS Responses.

The email feed uses the same username and password to connect to the receive and send email servers so that the entity receiving the email contacts is the same entity replying to the email contacts.

DELETE

Deletes a feed from the database.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/feed/ID Variables
HTTPS method:	DELETE
HTTPS response headers:	See HTTPS Responses.

GET (List)

Retrieves a list of all feeds in the system.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/feed?summary=<true/false>
HTTPS method:	GET
Parameters:	See Feed API Parameters, on page 94.
Example:	`https://<ServerIP>:<Port>/ccp-webapp/ccp/feed?summary=false`

Example XML response:	`<?xml version="1.0" encoding="UTF-8" standalone="yes"?>`
	`<feeds>`
	`  <Feed>`
	`  <changeStamp>2</changeStamp>`
	`  <chatInactivityTimeout>300</chatInactivityTimeout>`
	`  <chatJoinTimeout>60</chatJoinTimeout>`
	`  <description>My web chat feed</description>`
	`  <name>My Chat Feed</name>`
	`  <pushFeedURL>https://[Server]:[Port]/ccp-webapp/ccp/chatfeed/100159`
	
	`  </pushFeedURL>`
	`  <refURL>https://[Server]:[Port]/ccp-webapp/ccp/feed/100159</refURL>`
	
	`  <replyTemplateRefURL>`
	`  https://[Server]:[Port]/ccp-webapp/ccp/template/reply/302`
	`  </replyTemplateRefURL>`

```
        <status>1</status>
        <tags>
            <tag>tag1</tag>
            <tag>tag2</tag>
            <tag>tag3</tag>
        </tags>
        <type>8</type>
    </Feed>
    <Feed>
        <changeStamp>0</changeStamp>
        <description>callback</description>
        <name>callback</name>

<pushFeedURL>http://[Server]:[Port]/ccp/callback/feed/100012</pushFeedURL>
        <refURL>http://[Server]:[Port]/ccp-webapp/ccp/feed/100012</refURL>
        <status>1</status>
        <tags/>
        <type>10</type>
    </Feed>
    <Feed>
        <changeStamp>0</changeStamp>
        <description>My Task Feed</description>
        <name>My Task Feed</name>

<pushFeedURL>https://[Server]:[Port]/ccp/task/feed/100116</pushFeedURL>
       <refURL>https://[Server]:[Port]/ccp-webapp/ccp/feed/100116</refURL>

        <status>1</status>
        <tags>
            <tag>tag1</tag>
            <tag>tag2</tag>
            <tag>tag3</tag>
        </tags>
        <type>12</type>
    </Feed>
    <Feed>
        <changeStamp>4</changeStamp>
        <description>My Push Feed</description>
        <name>My Push Feed</name>
        <pushFeedURL>http://[Server]:[Port]/ccp-webapp/ccp/pushfeed/100019
          </pushFeedURL>
        <refURL>http://[Server]:[Port]/ccp-webapp/ccp/feed/100019</refURL>
        <status>1</status>
        <tags>
            <tag>pushfeed</tag>
            <tag>some_tag</tag>
        </tags>
        <type>7</type>
    </Feed>
    <Feed>
        <name>My Email Feed</name>
        <description>My Email Feed</description>
        <email>
            <receive>
                <host>imap.email.com</host>
                <port>993</port>
                <folderName>Inbox</folderName>
                <snapshotAge>120</snapshotAge>
            </receive>
            <send>
                <host>smtp.email.com</host>
                <port>587</port>
            </send>
            <username>me@email.com</username>
```

	<password>******</password> </email> <pollingInterval>60</pollingInterval> <replyTemplateRefURL>https://[Server]: [Port]/ccp-webapp/ccp/template/reply/105678</replyTemplateRefURL> <tags> <tag>tag1</tag> <tag>tag2</tag> </tags> <type>11</type> </Feed> </feeds>
HTTPS response headers:	https/1.1 200 OK Content-Type: application/xml Transfer-Encoding: chunked Date: Tue, 12 Jan 2010 16:47:58 GMT See also HTTPS Responses.

GET

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

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/feed/<id> For more information about <id>, see ID Variables, on page 2.
HTTPS method:	GET
Example XML response:	<Feed> <changeStamp>2</changeStamp> <chatInactivityTimeout>300</chatInactivityTimeout> <chatJoinTimeout>60</chatJoinTimeout> <description>My web chat feed</description> <name>My Chat Feed</name> <pushFeedURL>https://[Server]:[Port]/ccp-webapp/ccp/chatfeed/100159 </pushFeedURL> <refURL>https://[Server]:[Port]/ccp-webapp/ccp/feed/100159 </refURL> <replyTemplateRefURL> https://[Server]:[Port]/ccp-webapp/ccp/template/reply/302 </replyTemplateRefURL> <status>1</status> <tags> <tag>tag1</tag> <tag>tag2</tag> <tag>tag3</tag> </tags> <type>8</type> </Feed>
HTTPS response headers:	https/1.1 200 OK Content-Type: application/xml Transfer-Encoding: chunked Date: Tue, 12 Jan 2010 16:50:46 GMT

	See also HTTPS Responses.

PUT

Updates an existing feed.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/feed/<id> For more information about <id>, see ID Variables, on page 2.
HTTPS method:	PUT
Parameters:	See Feed API Parameters, on page 94.
Example XML request payload:	<Feed> <changeStamp>2</changeStamp> <chatInactivityTimeout>300</chatInactivityTimeout> <chatJoinTimeout>60</chatJoinTimeout> <description>My web chat feed</description> <name>My Chat Feed</name> <pushFeedURL>https://[Server]:[Port]/ccp-webapp/ccp/chatfeed/100159</pushFeedURL> <refURL>https://[Server]:[Port]/ccp-webapp/ccp/feed/100159</refURL> <replyTemplateRefURL> https://[Server]:[Port]/ccp-webapp/ccp/template/reply/302 </replyTemplateRefURL> <status>1</status> <tags> <tag>tag1</tag> <tag>tag2</tag> <tag>tag3</tag> </tags> <type>8</type> </Feed>
HTTPS response headers:	https/1.1 200 OK Content-Type: text/plain Content-Length: 0 Date: Thu, 14 Jan 2010 15:49:17 GMT See also HTTPS Responses.

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.

Note Email feeds contain parameters not used by other feed types. For information about parameters for email feeds, see Email Feed API Parameters, on page 97.

Parameter name	Description	Notes
authenticationPassword	The password for the username provided for an account.	String
authenticationUsername	The username for an account.	String
authToken	This is the oAuth access token.	String
changeStamp	The change stamp of the feed record.	Integer. Defaults to 0. Required for PUT (update). Is returned in GET.
chatJoinTimeout	The amount of time (in seconds) that the agent has to join the chat room.	
chatInactivityTimeout	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.	
description	The description of the feed.	String
email	A collection of parameters specific to creating an email feed.	For information about these email-specific parameters, see Email Feed API Parameters, on page 97. For an example of an email feed, see POST (Create an IMAP Email Feed), on page 90.
minAge	The minimum post age in seconds (defaults to 0).	If a post is newer than the *minAge*, it will not be stored by the feed.
name	The name of the feed.	String, must be unique. Required for creating (POST).
pollingInterval	The amount of time in seconds the system waits between attempts to read this feed.	Integer
pushFeedURL	The URL to which you push the entities that will become social contacts.	
refURL	A copy of the URL requested.	Response for GET.
replyTemplateRefURL	The URL of the reply template used to respond to social contacts obtained from this feed.	String If this field is blank, no reply template is used.

Parameter name	Description	Notes
status	The authorization status for feed.	Values are: • AUTHENTICATION_ NONE=1
summary	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.
tags	Contacts coming in from this feed will automatically be tagged with these default tags.	String. A maximum of 10 tags are allowed.
type	The feed type.	Required for POST Integer, types are: • Push = 7 • Chat = 8 • Callback = 10 • Email = 11 • Task = 12
url	The location of the feed that you want to read.	String

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

Note This table does not include information about email feed parameters. For information about which email feed parameters are required or optional, see Email Feed API Parameters, on page 97.

Feed type/ Field name	Push (7)	Chat (8)	Callback (10)/Task (12)
type	**R**	**R**	**R**
name	**R**	**R**	**R**
description	**O**	**O**	**O**
url	NA	NA	NA
polling **interval**	NA	NA	NA

Feed type/ Field name	Push (7)	Chat (8)	Callback (10)/Task (12)
minAge	NA	NA	NA
authentication **Username**	NA	NA	NA
authentication **Password**	NA	NA	NA
keywords	NA	NA	NA
authToken	NA	NA	NA
chatJoin **Timeout**	NA	**O**	NA
chat **Inactivity** **Timeout**	NA	**O**	NA
reply **Template** **RefURL**	**O**	**O**	**O**
tags	**O**	**O**	**O**

Email Feed API Parameters

This table defines the parameters used by the Email Feed API.

Parameter name	Description	Notes
type	The feed type.	Required. The following values are allowed: EMAIL - 11
name	The name of the feed.	Required.
description	The description of the email feed.	Optional.
email	Configuration information specific to email.	Required. This parameter is specific to email feeds.

Parameter name	Description	Notes
receive	Configuration information specific to receiving email.	Required. This parameter is specific to email feeds.
-->host	The hostname or IP address of the IMAP server.	Required. This parameter is specific to email feeds.
-->port	The port of the IMAP server.	Required. The default value is 993. This parameter is specific to email feeds.

-->folderName	The name of the folder from which to fetch the email.	Required.
		The default value is Inbox.
		This folder cannot be a shared or public folder.
		This parameter is specific to email feeds.
		In an Exchange mailbox, you can configure multiple folders. You can also configure folders within folders. For example, your email folders may be set up like the following:
		`Inbox`
		`Important Mail`
		`  Sales`
		`  Service`
		`Spam`
		In this example, if you want to configure an email feed to fetch mail from the Sales folder, you must include both the Important Mail folder and the Sales folder in the folderName parameter.
		`<folderName>Important Mail/Sales</folderName>`
		If the Sales folder includes two folders (Product A and Product B) and you want to configure an email feed to fetch mail from the Product A folder, set the folderName parameter as follows:
		`<folderName>Important Mail/Sales/Product A</folderName>`

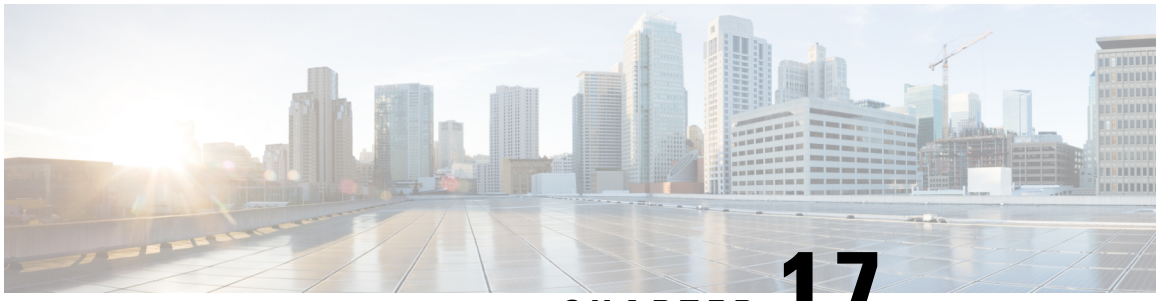
-->snapshotAge	The length of time (in minutes) that the email feed goes back to retrieve the initial set of email contacts.	Required. The default value is 120 minutes (2 hours). That is, by default, an email feed goes back 2 hours on the initial fetch. This parameter is specific to email feeds.

Parameter name	Description	Notes
send	Configuration information specific to sending email.	Required. This parameter is specific to email feeds.
-->host	The hostname or IP address of the SMTP server.	Required. This parameter is specific to email feeds.
-->port	The port of the SMTP server.	Required. The default value is 587. This parameter is specific to email feeds.
username	The email address used to connect to the email host.	Required. This parameter is specific to email feeds.
password	The password used to connect to the email host.	Required. This parameter is specific to email feeds.
pollingInterval	The amount of time (in seconds) that the system waits between attempts to fetch email.	Required.
replyTemplateUrl	The URL of the reply template used to respond to email contacts obtained from this feed.	Optional.
tags	Contacts coming in from this feed are automatically tagged with these default tags.	A maximum of 10 tags are allowed.

Email Feed Limitations

- SocialMiner does not verify the SSL certificate for the IMAP or SMTP server when establishing a connection.
- SocialMiner does not block spam when retrieving email messages. SocialMiner creates a social contact for each email that arrives in the configured user's folder.
- SocialMiner requires that the combination of receive host, receive folderName, and username for an email feed are unique. That is, no two email feeds can have the same values for all three parameters.

  For example, if two email feeds have the same values for receive host and receive folderName, the value for username must be different for each feed.



CHAPTER 17

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, on page 101
- About Script Filters, on page 106

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 101
Delete, on page 102
Get List, on page 102
Get, on page 103
GET (Script Content), on page 104
Put, on page 104
Filter API Parameters, on page 105

Post

Creates a filter to be stored in the database.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/filter
HTTPS method:	POST
Parameters:	See Filter API Parameters, on page 105.

Example XML request payload for Bayesian filter:	`<Filter>`
	`<name>Bayesian</name>`
	`<description>Bayesian</description>`

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

Delete

Removes a filter from the database.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/filter/<ID Variables>
HTTPS method:	DELETE
HTTPS response headers:	https/1.1 200 OK Content-Type: text/plain Content-Length: 0 Date: Tue, 12 Jan 2010 17:03:54 GMT

Get List

Returns a list of all filters.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/filter
HTTPS method:	GET

URL parameter:	True or false. Defaults to false. When “true”, only the URLs of the objects are returned. When “false”, full object information is returned along with the URL reference.
Parameters:	See Filter API Parameters, on page 105.
Example:	`https://<ServerIP>:<Port>/ccp-webapp/ccp/filter?summary=false`

Example XML response :	`<Filters>`
	`  <Filter>`
	`  <changeStamp>0</changeStamp>`
	`  <description>this is an AUTHOR filter!</</description>`
	`  <keywords>`
	`  <keyword>author name</keyword>`
	`  <keyword>author userid</keyword>`
	`  </keywords>`
	`  <name>filter1</name>`
	`  <refURL>`
	`  https://[ServerIP]:[Port]/ccp-webapp/ccp/filter/[id]`
	`  </refURL>`
	`  <rule>1</rule>`
	`  <systemDefined>false</systemDefined>`
	`  <type>3</type>`
	`  </Filter>`
	`  <Filter>`
	`  <changeStamp>0</changeStamp>`
	`  <description>`
	`  System defined filter for Context Service Integration.`
	`  </description>`
	`  <name>Context Service Integration Filter</name>`
	`  <refURL>`
	`  https://[ServerIP]:[Port]/ccp-webapp/ccp/filter/90001`
	`  </refURL>`
	`  <scriptContentRefURL>`
	`  https://[ServerIP]:[Port]/ccp-webapp/ccp/filter/90001`
	`  /scriptcontent`
	`  </scriptContentRefURL>>`
	`  <scriptFileName>`
	`  ContextServiceIntegrationFilter.groovy`
	`  </scriptFileName>`
	`  <systemDefined>true</systemDefined>`
	`  <type>4</type>`
	`  </Filter>`
	`  </Filters>`

HTTPS response headers:	`https/1.1 200 OK`
	`Content-Type: application/xml`
	`Transfer-Encoding: chunked`
	`Date: Tue, 24 Jun 2015 16:47:58 GMT`

Get

Returns the data for a single filter.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/filter/<ID Variables>
HTTPS method:	GET

Example:	`https://<ServerIP>:<Port>/ccp-webapp/ccp/filter/100036`
Parameters:	See Filter API Parameters, on page 105.

Example XML response for a script filter:	`<?xml version="1.0" encoding="UTF-8" standalone="yes"?>`
	`<Filter>`
	`<changeStamp>0</changeStamp>`
	`<keywords/>`
	`<name>Filter name</name>`
	`<refURL>`
	`https://[ServerIP]:[Port]/ccp-webapp/ccp/filter/100000`
	`</refURL>`
	`<scriptContentRefURL>`
	`https://[ServerIP]:[Port]/ccp-webapp/ccp/filter/100000`
	`/scriptcontent`
	`</scriptContentRefURL>`
	`<scriptFileName>me.groovy</scriptFileName>`
	`<systemDefined>false</systemDefined>`
	`<type>4</type>`
	`</Filter>`

HTTPS response headers:	`https/1.1 200 OK`
	`Content-Type: application/xml`
	`Date: Tue, 23 Jun 2015 17:40:46 GMT`

GET (Script Content)

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

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/filter/(id)/scriptcontent
HTTPS method:	GET

Response for a script filter:	`function{`
	`if((a<b) && (c>d) ) {`
	`after=before;`
	`before=after;`
	`}`
	`}`

Put

Updates an existing filter.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/filter/<ID Variables>
HTTPS method:	PUT
Parameters:	See Filter API Parameters, on page 105.

Example XML request payload for Bayesian filter:	`<Filter>`
	`<name>Bayesian2</name>`
	`<description>a different description</description>`
	`<changeStamp>0</changeStamp> <type>2</type>`
	`</Filter>`

Example XML request payload for author filter:	<Filter> <name>String</name> <description>String</description> <type>3</type> <changeStamp>12345</changeStamp> <keywords> <keyword>author name</keyword> <keyword>author userid</keyword> <keyword>alternate form on author name</keyword> </keywords> <rule>1</rule> </Filter>
Example XML request payload for script filter:	<Filter> <changeStamp>12345</changeStamp> <name>String</name> <description>A new description</description> <type>4</type> <scriptFileName>me.groovy</scriptFileName> <scriptContent> <![CDATA[put the updated text of the script here.]]> </scriptContent> </Filter>
HTTPS response headers:	https/1.1 200 OK Content-Type: text/plain Content-Length: 0 Date: Thu, 14 Jan 2010 15:49:17 GMT

Filter API Parameters

Parameters are optional unless otherwise noted.

Parameter	Description	Notes
name	The name of the Context Service Filter. Must be unique.	Required for POST. Maximum of 255 characters.
description	A description of the Context Service Filter.	Null = yes
changeStamp	The change stamp of the filter record. Integer. Defaults to 0.	A changeStamp is required for PUT (update API). changeStamp is returned in GET.
lastupdated	Last time this row was updated.	Integer.
ouid	tbd	Integer.
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. Required for POST (create) for script filters.

Parameter	Description	Notes
systemDefined	True if the filter was preinstalled on SocialMiner.	POST, PUT and DELETE operations ignore this parameter.
scriptFileName	Should be the same as the uploaded script filter file name.	Used only for script filters. Required for POST (create) for script filters.
type	The type of filter. Must be one of: • 0 = UNKNOWN • 1 = WORD_COUNT_ LESS_THAN_SIX • 2 = Bayesian • 3 = Author • 4 = SCRIPT	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.

Note You cannot add, update or delete the system-defined filter.

Related Topics

Script Binding, on page 106
Develop and Test Script Filters, on page 107
Script Filter Security, on page 110
Sample Script Filters, on page 111

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 25 to obtain a social contact link. Each <entry> has a <link rel="socialcontact" href="socialcontactRefURL">.

Step 2 Run a Filters Get List, on page 102 to obtain the refURL of the script filter you want to test.

Step 3 Run a Filter Results GET, on page 113 and add the refURL of the social contact as a parameter.

Example:

```
https://<ServerIP>:<Port>/ccp-webapp/ccp/filter/
103105/results?socialContact=
https://<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>
 https://[ServerIP]:[Port]/ccp-webapp/ccp/
 filter/103105/results</refURL>
<result>100</result>
<socialContact>
 https://[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/
 AB1C351410000132000000F450A568DD2
</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

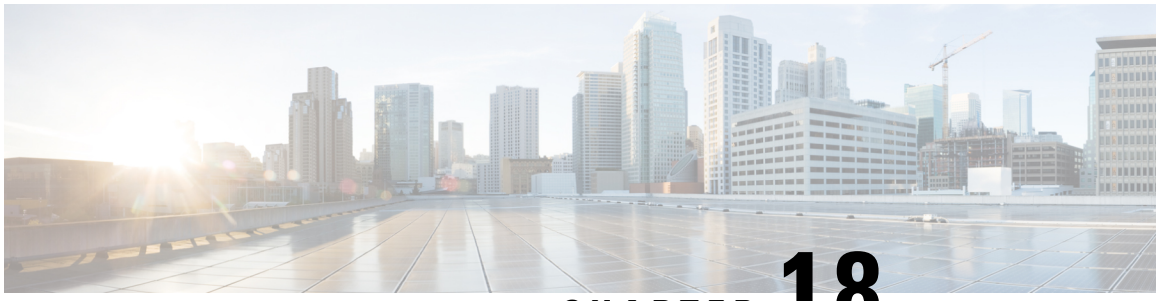
You can create script filters to change or add to the content of social contacts and to call external web services. For example, you can create a script to translate text to another language, to analyze sentiment (opendover), or to recognize trends (Google Prediction).

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 ->
https://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
```



CHAPTER 18

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, on page 113

Filter Results API Commands

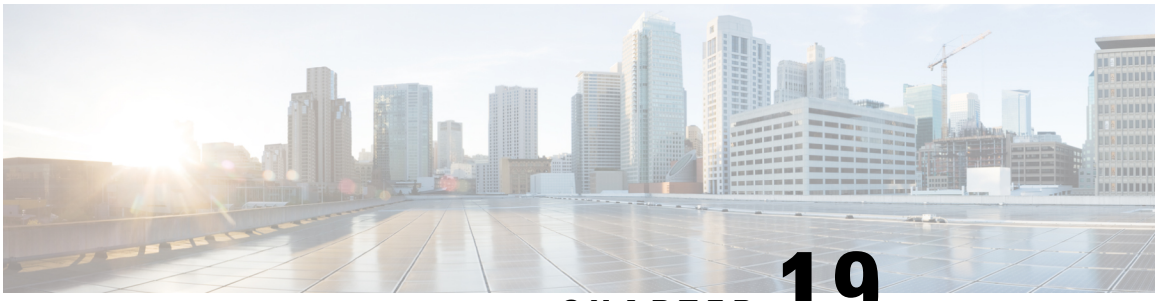
This section describes the supported command (GET) for the filter results API and the parameters for that command.

GET

Get results for the specified filter.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/filter/<ID Variables>/results
HTTPS method:	GET
URL parameters:	• **document**: the text that the configured filter analyzes. • **socialContact:** the refURL of the social contact that the configured filter analyzes. It is valid to provide either document or social contact, but invalid to provide both or neither.
Example XML response:	The filter results are returned as a single <FilterResult> element that contains five required child elements. **FilterResult**: the container for the result. • **document**: the text passed to the filter for analysis. • **logBuffer**: the aggregation of the social contact filter log output. • **refURL**: the URL of the filter results request. • **result**: the result of the filter analysis expressed as an integer from 1–100. • **socialContact**: the refURL of the social contact that was passed to the filter for analysis. <FilterResult> <refURL> https://[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>
```



CHAPTER 19

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	

Rule type	Required parameters	Optional parameters	Notes
cce	For Callback feed name, campaignUrl, tags, type, scriptSelector, mediaRoutingDomainId	description	The scriptSelector is the Dialed Number String or Script Selector from the CCE configuration. The mediaRoutingDomainId is the ID of the selected Media Routing Domain from the CCE configuration. For voice callback, the mediaRoutingDomainId is 1.
	For Request feed name, campaignUrl, tags, type,	description, scriptSelector, mediaRoutingDomainId	Do not specify the mediaRoutingDomainId; -1 is set as the default value for request. Do not specify the scriptSelector; value is picked from the Request.

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, on page 116
- Notification Keywords in Email and IM , on page 122
- HTTPS Notifications, on page 122

Notification API Commands

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

POST

Creates a notification rule.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/notificationrule
HTTPS method:	POST
Example XML request payload (email):	<NotificationRule> <name>test</name> <description>this is the description</description> <campaignUrl>http://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/

	MyTestCampaign</campaignUrl> <tags> <tag>test</tag> <tag>cisco</tag> </tags> <targets> <target>test@cisco.com</target> <target>cisco@cisco.com</target> </targets> <type>email</type> <subject>Notification Rules</subject> <body>Click on this link.</body> </NotificationRule>
Example XML request payload (http):	<NotificationRule> <name>test</name> <description>this is the description</description> <campaignUrl>http://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/MyTestCampaign</campaignUrl> <tags> <tag>test</tag> <tag>cisco</tag> </tags> <type>http</type> <httpUrl>http://someserver/notification/handler</httpUrl> <httpUsername>username</httpUsername> <httpPassword>password</httpPassword> <sslVerifyCertificates>true</sslVerifyCertificates> </NotificationRule>
Example XML request payload for Callback (CCE):	<NotificationRule> <name>test</name> <description>this is the description</description> <campaignUrl>http://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/MyTestCampaign</campaignUrl> <tags> <tag>test</tag> <tag>cisco</tag> </tags> <type>cce</type> <scriptSelector>allSalesAndService</scriptSelector> <mediaRoutingDomains> <mediaRoutingDomain> <mediaRoutingDomainId>1</mediaRoutingDomainId> </mediaRoutingDomain> </mediaRoutingDomains> </NotificationRule>
Example XML request payload for Task (CCE):	<NotificationRule> <description>this is the description</description> <campaignUrl>http://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/MyTestCampaign</campaignUrl> <tags> <tag>test</tag> <tag>cisco</tag> </tags> <type>cce</type> </NotificationRule>
Parameters:	See Notification API Parameters, on page 120.

HTTPS response headers:	A *201 Created* http header is returned on success, along with the REST URL to the new notification rule.

PUT

Updates an existing notification rule.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/notificationrule/<ID Variables>
HTTPS method:	PUT
Parameters:	See Notification API Parameters, on page 120. All parameters are optional for the Notification API update operation.
Example XML response:	<NotificationRule> <body>New Contact:</body> <campaignUrl> https://[ServerIP]:[Port]/ccp-webapp/ccp/ campaign/Pushed_Contacts </campaignUrl> <changeStamp>0</changeStamp> <name>Push</name> <subject> Notification: New Push Tag applied to Pushed Contacts Campaign </subject> <tags> <tag>push</tag> </tags> <targets> <target>user@example.com</target> </targets> <type>email</type>
HTTPS response headers:	A *200 OK* http header is returned on success.

DELETE

Deletes a notification rule.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/notificationrule/<ID Variables>
HTTPS method:	DELETE
HTTPS response headers:	A *200 OK* http header is returned on success.

GET (List)

Lists all notification rules.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/notificationrule

HTTPS method:	GET
Example XML response:	<NotificationRules> <NotificationRule> <body>New Contact:</body> <campaignUrl> https://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/ Pushed_Contacts </campaignUrl> <changeStamp>0</changeStamp> <name>Push</name> <refURL> https://[ServerIP]:[Port]/ccp-webapp/ccp/ notificationrule/100010 </refURL> <subject> Notification: New Push Tag applied to Pushed Contacts Campaign </subject> <tags> <tag>push</tag> </tags> <targets> <target>user@example.com</target> </targets> <type>email</type> </NotificationRule> <NotificationRule> .... </NotifactionRule> </NotificationRules>
HTTPS response headers:	A *200 OK* http header is returned on success.

GET

Retrieves a specific notification rule.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/notificationrule/<ID Variables>
HTTPS method:	GET
Example XML response:	<NotificationRule> <body>New Contact:</body> <campaignUrl>https://[ServerIP]:[Port]/ ccp-webapp/ccp/campaign/Pushed_Contacts</campaignUrl> <changeStamp>0</changeStamp> <name>Push</name> <refURL>https://[ServerIP]:[Port]/ ccp-webapp/ccp/notificationrule/100010</refURL> <subject>Notification: New Push Tag applied to Pushed Contacts Campaign</subject> <tags> <tag>push</tag> </tags> <targets> <target>user@example.com</target>

	<pre></targets>
	<type>email</type>
	</NotificationRule></pre>

HTTPS response headers:	A *200 OK* 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
changeStamp	The change stamp of the notification record.	Integer. Defaults to 0. changeStamp is returned in GET. changeStamp, on page 3 is required for PUT (Update API).
name	The name of the notification rule.	Required for POST.
campaignUrl	The URL of the campaign.	String. Required for POST.
description	Description.	String. Follows SocialMiner standard description naming conventions.
tags/tag	Tag or list of tags from which the rule is activated and a notification is sent.	String. Required for POST. Maximum of 5.
type	The type of notification to send out for this rule.	Valid values are: • *email* (Notification is sent over email.) • *IM* (Notification is sent over IM.) • *http* (Notification is used for chat and is reserved for other developer applications that deliver social contact data to an external application.) • *(connection to) CCE* (Notification sends a request to CCE with media routing information.) Required for POST.

Parameter	Description	Notes
targets/target	One or more targets to which the notification rule is sent.	String. Required for POST. Maximum of 10.
httpUrl	The URL of a REST API.	Required for POST. SocialMiner will post details of a social contact to this URL.
subject	The subject of a notification rule message.	String. Maximum of 255 characters.
body	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. Maximum of 2048 characters. The body may contain reserved keywords in a special syntax. These Notification Keywords in Email and IM will be replaced with values from the social contact. The reserved word syntax takes the form ${KEYWORD}.
httpUsername	Username	Required if authentication is necessary to use the REST API specified by **httpUrl**.
httpPassword	Password	Required if authentication is necessary to use the REST API specified by **httpUrl**.
sslVerifyCertificates	Defines if SSL certificate verification will be enabled or disabled for connections made for notification.	Boolean. Default is True. HTTPS only.
scriptSelector	The Dialed Number String/Script Selector from the CCE configuration.	String. Maximum of 32 characters.
mediaRoutingDomains	A list of media routing domains, each of which contains the media routing domain ID of the media routing domain from the CCE Configuration.	Positive integer.

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.
- 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.

HTTPS Notifications

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

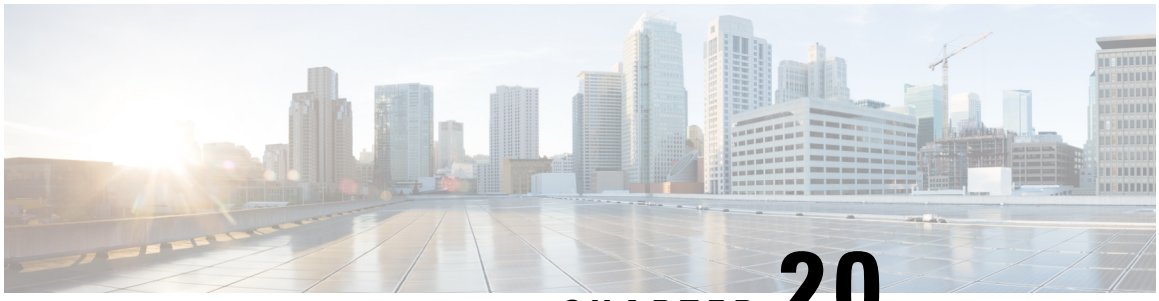
HTTPS method:	POST
Fields:	• **author**: author of the social contact.

	• **description**: body of the social contact.
	• **id**: datastore id of the social contact.
	• **link**: unique id of the original social contact.
	• **notificationTag**: the tag that fired off the notification rule.
	• **publishedDate**: the publish date of the document.
	• **refURL**: the REST id of the social contact. Applications can do a GET on this URL to get social contact detail.
	• **screenPopUrl**: URL used to access the social contact in the SocialMiner's UI.
	• **sourceType**: the type of feed to which the social contact belongs. Valid values are
	  • chat
	  • push
	  • callback
	  • email
	• **status**: string (case-sensitive) One of:
	  • unread: The default state of a new social contact.
	  • reserved: Reserved to be handled.
	  • handled: This social contact has been handled and no further action is required.
	  • discarded: This social contact does not require a response and is filed in the recycle bin.
	  • queued: The callback request was successfully submitted to the contact center for routing.
	• **statusTimestamp**: timestamp of the last status update.
	• **statusUserId**: id of the agent who updated the status (initially this is empty).
	• **statusReason**: the reason the contact was moved to the current status.
	• **tags/tag**: one or more tags associated with the social contact—normally this is optional but for CCX integration this should contain the routing tag.
	• **extensionFields/extensionField**: 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.
	• **title**: title of the social contact.
	• **replyTemplateRefURL:** link to the reply template configuration. Present if reply template is configured for the contact.
	• **replyTemplateURL:** link to the reply template. Present if reply template is configured for the contact.
	• **replyType:** expected reply format (currently web, chat, or email). The URI below will be in a format compatible with the reply type.
	• **URI:** for chat room.

Example https notification message:	`<SocialContact>`
	`<author>Michael Littlefoot</author>`
	`<description>Web Chat</description>`
	`<extensionFields>`
	`<extensionField>`
	`<name>h_Name</name>`

```
            <value>Michael Littlefoot</value>
        </extensionField>
        <extensionField>
            <name>h_chatOrigin</name>
            <value>FACEBOOK_MESSENGER</value>
        </extensionField>
        <extensionField>
            <name>h_chatRoom</name>
            <value>socialminer_chat.0@conference.127.0.0.1</value>
        </extensionField>
    </extensionFields>
    <id>7ED5E7C5100001640000000000A4E5F26</id>
    <integrationAuthTokenExpirationTimestamp>1531157905093
    </integrationAuthTokenExpirationTimestamp>
    <integrationAuthTokenGUID>XXXXXXXXX</integrationAuthTokenGUID>
    <link>https://[Server]:[Port]/ccp/socialcontact/
     7ED5E7C5100001640000000000A4E5F26 /link>
    <notificationTag>ccx_chat_req</notificationTag>
    <publishedDate>1531136305092</publishedDate>
    <refURL>https://[Server]:[Port]/ccp-webapp/ccp/socialcontact/
     7ED5E7C5100001640000000000A4E5F26</refURL>
    <replyTemplateURL>https://[Server]:[Port]/multisession/ui/
    templates/reply/cisco_agent_ms_chat.jsp</replyTemplateURL>
    <screenPopUrl>https://[Server]:[Port]/results.jsp?scID=
    \7ED5E7C5100001640000000000A4E5F26&amp;campaignID=
    CCX_Chat_Campaign</screenPopUrl>
    <sourceType>chat</sourceType>
    <status>unread</status>
    <statusReason>unknown</statusReason>
    <statusTimestamp>1531136305095</statusTimestamp>
    <statusUserId></statusUserId>
    <tags>
        <tag>ccx_chat_req</tag>
    </tags>
    <title>Chat from Facebook Messenger</title>
</SocialContact>
```


HTTPS response headers:	A *200 OK* http header is returned on success.



CHAPTER 20

Predefined Response

The predefined response API is used to define a set of predefined responses to common questions that agents can use when replying to chat or email contacts.

- Create Predefined Response, on page 125
- Get Predefined Response, on page 127
- List Predefined Response, on page 128
- List Predefined Response (By Group), on page 129
- List Predefined Response (By Type), on page 130
- List Predefined Response(Sorted by Title), on page 130
- Update Predefined Response, on page 130
- Delete Predefined Response, on page 131

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.
- The value of the contentType parameter can be up to 85 characters.
- At least one group name has to be specified in a response.
- Groups are classified into two, namely:**response**(default) and **signature.**

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/predefinedResponse/
HTTPS method:	POST

Example XML response (plain text predefined response):	<response> <title>A summary of the response</title> <text>The text of the response. This can be up to 2000 utf8 characters long</text> <type>response</type> <groups> <group>group 1</group> <group>group 2</group> <group>group 3</group> </groups> </response>
Example XML request payload (HTML type predefined response):	<response> <title>A summary of the response</title> <text><![CDATA[<!DOCTYPE html><html><head><title>This is a sample HTML response </title></head><body> My First HTML </body></html>]]></text> <contentType>text/html</contentType> <type>response</type> <groups> <group>group 1</group> <group>group 2</group> <group>group 3</group> <group>group n</group> </groups> </response>
Parameters:	**title** (required) : a short summary of the response. **text** (required) : the text of the response. If the text content is HTML, you must place the HTML text inside a CDATA section (<![CDATA[...html...]]>). **contentType** (required for email/HTML predefined responses, optional for others): the type of text included in the response. The value for this parameter is freeform. For example, you could use a value of text/plain when the response text includes unformatted, readable text (such as for chat predefined responses). You must use a value of text/html for email predefined responses when the response text includes information formatted with html tags. **type** (optional): the type of the template. This can have two values: • **response**: Indicates a response template type. This is the default value when the parameter is not specified. • **signature**: Indicates a signature template type which is automatically appended to the end of outgoing emails. **groups** (required) : the list of groups associated with the response.
HTTPS response headers:	201 Created (the created URL is returned with the response) 400 Bad request (if the input is not as per the defined criteria) For more information, see HTTPS Responses.

Note In predefined responses of type **signature**, a specific variable `${agent_alias}` can be inserted as part of the text attribute. All occurrences of this variable will automatically be replaced with the actual alias of the Finesse agent when the signature is added to an email reply from the multi-session reply gadget. In cases where the agent does not have an alias configured, all occurrences of this variable will be replaced with the agent ID instead.

Get Predefined Response

This API retrieves a predefined response from the system.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/predefinedResponse/<id>
HTTPS method:	GET
Example XML response (plain text predefined response):	<response> <id>The GUID generated by the API for this response</id> <changeStamp>2</changeStamp> <title>The title of the response.</title> <text>The text of the response.</text> <contentType>text/plain</contentType> <refURL>http://<server>:<serverport>/ccp-webapp/ccp/response/(id)</refURL> <groups> <group>group 1</group> <group>group 2</group> <group>group 3</group> <group>group n</group> </groups> </response>

Example XML response (HTML predefined response):	<response>
	  <id>The GUID generated by the API for this response</id>
	  <changeStamp>2</changeStamp>
	  <title>A 100 utf8 character title</title>
	  <text><![CDATA[<!DOCTYPE
	html><html><head><title>
	  This is a sample HTML</title></head><body>
	
	  My First HTML </body></html>]]></text>
	  <contentType>text/html</contentType>
	
	<refURL>http://<server>:<serverport>/ccp-webapp/ccp/response/(id)</refURL>
	
	  <groups>
	  <group>group 1</group>
	  <group>group 2</group>
	  <group>group 3</group>
	  <group>group n</group>
	  </groups>
	</response>

Elements:	**title** : a short summary of the response. **text** : the text of the response. If the text is HTML, the content is XML-encoded **contentType**: the type of text included in the response. **type** (optional): the type of the template. This can have two values: • **response**: Indicates a response template type. This is the default value when the parameter is not specified. • **signature**: Indicates a signature template type which is automatically appended to the end of outgoing emails. **groups** : the list of groups associated with the response. **id** : contains the GUID identifying a predefined response.
HTTPS response headers:	200 OK 404 Not found For more information, see HTTPS Responses.

List Predefined Response

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

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/predefinedResponse
HTTPS method:	GET

Example XML response:	<responses> <response> <id>The GUID generated by the API for this response</id> <title>Title</title> <text>The text of the response.</text> <type>response</type> <refURL>https://<server>:<serverport>/ccp-webapp/ccp/response/(id)</refURL> <groups> <group>group 1</group> <group>group 2</group> </groups> </response> <response> <id>The GUID generated by the API for this response</id> <title>Title</title> <text>Another text of a response. </text> <type>signature</type> <refURL>https://<server>:<serverport>/ccp-webapp/ccp/response/(id)</refURL> <groups> <group>group 3</group> <group>group n</group> </groups> </response> <response> <id>The GUID generated by the API for this response</id> <title>Another 100 utf16 character title</title> <text><![CDATA[<!DOCTYPE html><html>< head><title>This is a sample HTML</title></head> <body> My First HTML </body>< /html>]]></text> <contentType>text/html</contentType> <refURL>https://<server>:<serverport>/ccp-webapp/ccp/response/(id)</refURL> <groups> <group>group 3</group> <group>group n</group> </groups> </response> </responses>
HTTPS response headers:	200 OK For more information, see HTTPS Responses.

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**.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/predefinedResponse?group=<group1>&group=<group2>
HTTPS method:	GET
HTTPS response headers:	200 OK 400 Bad request For more information, see HTTPS Responses.

List Predefined Response (By Type)

This API retrieves a set of predefined responses by types: **response** or **signature**.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/predefinedResponse?type=<response | signature>
HTTPS method:	GET
HTTPS response headers:	200 OK 400 Bad request For more information, see HTTPS Responses.

List Predefined Response(Sorted by Title)

This API retrieves a set of predefined responses in case-sensitive, alphabetically sorted order of the "Title" field.

URL:	https://<serverIP>:<Port>/ccp-webapp/ccp/predefinedResponse?sorted=true
HTTPS method:	GET
HTTPS response headers:	200 OK 400 Bad RESPONSE For more information, see HTTPS Responses.

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

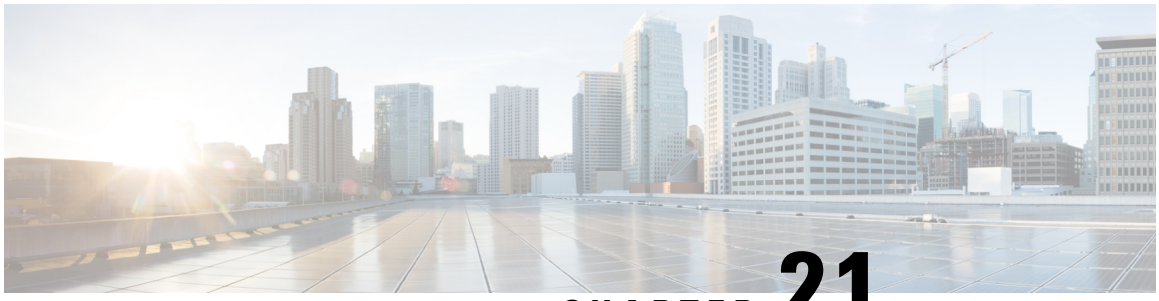
- Any field updated in the request replaces the existing values stored in the database.
- The *changeStamp* provided by the server should not be altered by the clients, and must be sent in the UPDATE request with other fields to be updated.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/predefinedResponse/<id>
HTTPS method:	PUT
Example XML request payload:	<response> <title>A summary of the response</title> <changeStamp>3</changeStamp> <text>The text of the response.</text> <contentType>text/plain</contentType> <groups> <group>group 1</group> <group>group 2</group> <group>group 3</group> </groups> </response> <response> <title>A summary of the response</title> <changeStamp>3</changeStamp> <text>The text of the response.</text> </response>
HTTPS response headers:	200 OK 400 Bad request For more information, see HTTPS Responses.

Delete Predefined Response

This API deletes a predefined response.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/predefinedResponse/<id>
HTTPS method:	DELETE
HTTPS response headers:	200 OK 400 Bad request For more information, see HTTPS Responses, on page 4.



CHAPTER 21

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, on page 133

Proxy API Commands

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

Related Topics

GET, on page 133
PUT, on page 134
Proxy API Parameters, on page 134

GET

Retrieves the proxy configuration.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/proxy/default
HTTPS 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> https://[ServerIP]:[Port]/ccp-webapp/ccp/

	proxy/default </refURL> </Proxy>

PUT

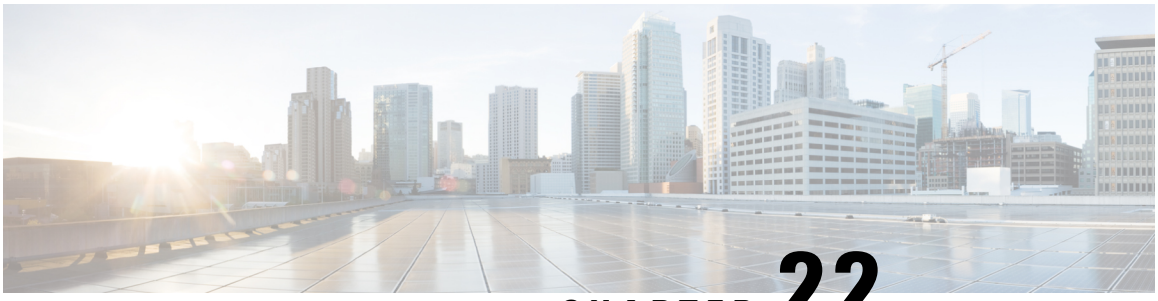
Updates the proxy configuration. By default, the configuration is blank and disabled.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/proxy/default
HTTPS method:	PUT
Parameters:	See Proxy API Parameters, on page 134.
Example XML request payload:	<Proxy> <host>[ServerIP]</host> <port>[port]</port> <exclusions> <exclusion>localhost</exclusion> <exclusion>*.cisco.com</exclusion> <exclusion>161.44.*</exclusion> <exclusion>192.168.1.1</exclusion> </exclusions> <enabled>true</enabled> </Proxy>

Proxy API Parameters

Parameters are optional unless otherwise noted.

Parameter	Description	Notes
enabled	True or false. Defines whether or not proxy use is enabled.	Defaults to false.
host	The fully-qualified hostname or IP address of the proxy server.	Required if enabled parameter is true.
port	The https port for the proxy server.	Required if enabled parameter is true.
exclusions	A list of host names to exclude from being used by the proxy.	The exclusion list is limited to 255 total characters (not including the <exclusion> tags). There is an additional character per item in the list that acts as a separator. Wildcards can be used. Examples: • localhost • *.cisco.com • xxx.yy.*



CHAPTER 22

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, on page 135
- PUT, on page 135

GET

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

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/publicprefix/default
HTTPS method:	GET
Example XML response:	<Publicprefix> <path> https://[public_server]:[serverport] [/optional/path] </path> <refURL> https://[ServerIP]:[Port]/ccp-webapp/ccp/ publicprefix/default </refURL> </Publicprefix> See also API Conventions, on page 1.

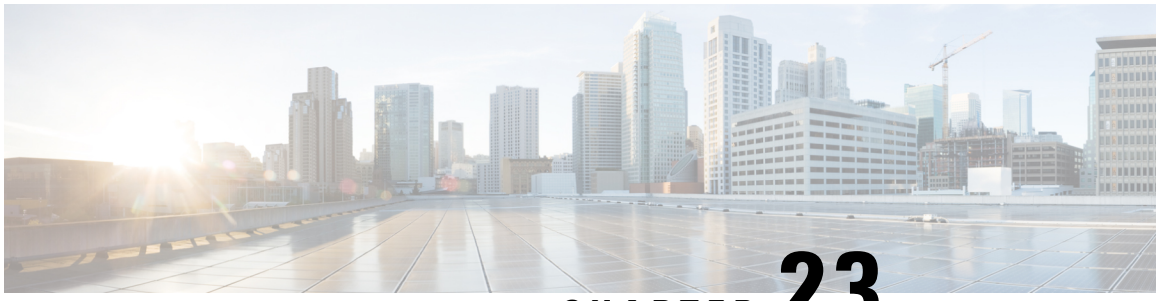
PUT

Updates the proxy configuration with the public prefix.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/publicprefix/default
HTTPS method:	PUT
Parameters:	**prefix**(optional)—specifies the public URL for the SocialMiner server and port. If no port number is provided, it defaults to 8000.

	If an empty prefix is provided (<Prefix><<prefix> </prefix></Prefix>), then the prefix will be set to nothing. The value for "public server" cannot be an IP address.

Example XML response:	`<Publicprefix>`
	`  <path>`
	`  https://[public_server]:[serverport]`
	`  [/optional/path]</path>`
	`</Publicprefix>`
	See also API Conventions, on page 1.



CHAPTER 23

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, on page 137
- Purge API Parameters, on page 138

Purge API Commands

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

Related Topics

GET (List), on page 137

PUT, on page 138

GET (List)

Lists the current purge settings.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/purge
HTTPS method:	GET
Parameters:	See Purge API Parameters, on page 138.
Example XML response:	<PurgeConfig> <dataStoreEmergencyPurgeDiskUsage> 50 </dataStoreEmergencyPurgeDiskUsage> <dataStorePurgeAge>30</dataStorePurgeAge> <reportingPurgeAge>550</reportingPurgeAge> <reportingPurgeTime>01:00</reportingPurgeTime> </PurgeConfig>

PUT

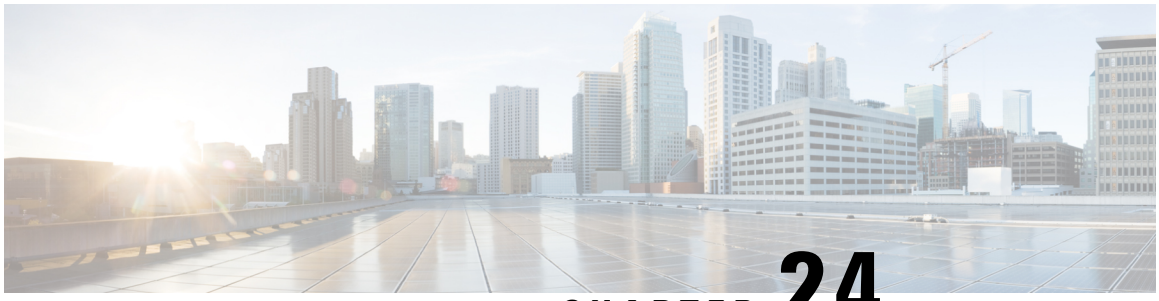
Updates the purge settings.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/purge
HTTPS method:	PUT
Parameters:	See Purge API Parameters, on page 138.
Example XML request payload:	<PurgeConfig> <dataStoreEmergencyPurgeDiskUsage> 80 </dataStoreEmergencyPurgeDiskUsage> <dataStorePurgeAge>30</dataStorePurgeAge> <reportingPurgeAge>550</reportingPurgeAge> <reportingPurgeTime>01:00</reportingPurgeTime> </PurgeConfig>
HTTPS response headers:	A *200 OK* http header is returned on success.

Purge API Parameters

All parameters are optional.

Parameter	Description	Notes
dataStoreEmergencyPurgeDiskUsage	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.
dataStorePurgeAge	The age of the social contacts that will be purged.	Must be a whole number between 1–550 (no decimal).
reportingPurgeAge	The time, in 24 hour format (HH:mm), when the purge is to start.	Valid values are 00:00 to 23:59.
reportingPurgeTime	The ages of reporting records that will be purged.	Valid values are 1 to 550.



CHAPTER 24

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, on page 139

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 139
GET, on page 139

POST

The social contact create (POST) method (POST, on page 173) is the preferred method for creating a social contact. Use the payload shown in that API.

GET

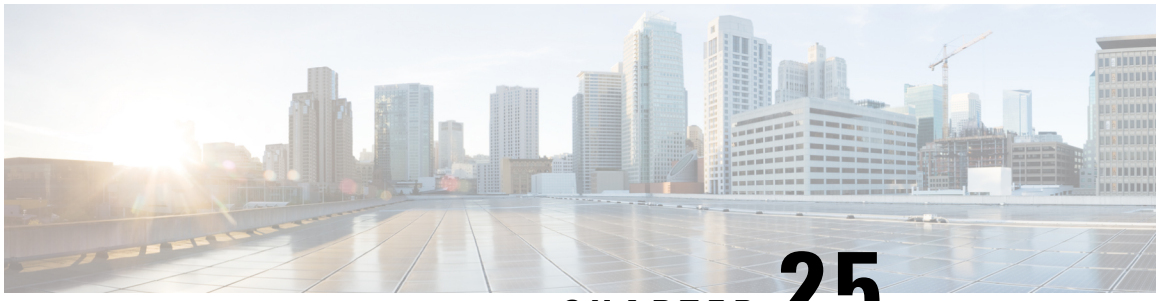
The GET method for push feeds is a simplified alternative to the social contact create (POST) API. It requires no API authentication.

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.)

An example of a push feed URL:	https://<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
HTTPS method:	GET
Parameters:	• **Title**—the title of the social contact. • **Author**—the author of the social contact. • **Description**—the content of the social contact. • **Tags**—a comma separated list of tags for this social contact. • **ExtensionField_<name>**—a custom name and value pair for this social contact. Tags and the extension field names cannot contain commas or colons.
HTTPS response headers:	A 200 response indicates success. See also HTTPS Responses.



CHAPTER 25

Reply Template API Commands

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

- POST, on page 141
- DELETE, on page 141
- GET (List), on page 142
- GET, on page 143
- PUT, on page 143
- Reply Template API Parameters, on page 143

POST

Creates a new template definition.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/template/reply
HTTPS method:	POST
Parameters:	See Reply Template API Parameters, on page 143.

Example XML request payload:	`<Template>`
	`<name>My  Template</name>`
	`<templateURL>`
	`http://this.is.my.template.url/template.html`
	`</templateURL>`
	`</Template>`

Response:	Status: 201 Created

DELETE

Deletes a custom reply template definition.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/template/reply/<ID Variables>

HTTPS method:	DELETE
HTTPS response headers:	Status 200: OK

GET (List)

Lists all custom reply templates stored on this system.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/template/reply
HTTPS method:	GET
Parameters:	See Reply Template API Parameters, on page 143.
Example response:	<templates> <Template> <changeStamp>0</changeStamp> <name>Cisco Chat</name> <refURL> http://[Server]:[Port]/ccp-webapp/ccp/template/reply/302 </refURL> <systemDefined>true</systemDefined> <templateURL>/templates/reply/cisco_agent_chat.jsp</templateURL> </Template> <Template> <changeStamp>0</changeStamp> <name>Cisco Email</name> <refURL> http://[Server]:[Port]/ccp-webapp/ccp/template/reply/304 </refURL> <systemDefined>true</systemDefined> <templateURL> /multisession/ui/templates/reply/cisco_agent_ms_email.jsp </templateURL> </Template> <Template> <changeStamp>0</changeStamp> <name>Cisco Multi-session Chat</name> <refURL> http://[Server]:[Port]/ccp-webapp/ccp/template/reply/303 </refURL> <systemDefined>true</systemDefined> <templateURL> /multisession/ui/templates/reply/cisco_agent_ms_chat.jsp </templateURL> </Template> <Template> <changeStamp>0</changeStamp> <name>My Reply Template 1</name> <refURL> http://[Server]:[Port]/ccp-webapp/ccp/template/reply/100201 </refURL> <systemDefined>false</systemDefined> <templateURL>https://example.com/templates/1</templateURL> </Template> <Template> <changeStamp>0</changeStamp> <name>My Reply Template 2</name>

	<refURL> http://[Server]:[Port]/ccp-webapp/ccp/template/reply/100202 </refURL> <systemDefined>false</systemDefined> <templateURL>https://example.com/templates/2</templateURL> </Template> </templates>

GET

Gets the details for one reply template.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/template/reply/<ID Variables>
HTTPS method:	GET
Example XML response:	<Template> <changeStamp>0</changeStamp> <name>My Reply Template 1</name> <refURL> http://[Server]:[Port]/ccp-webapp/ccp/template/reply/10020 </refURL> <systemDefined>false</systemDefined> <templateURL>https://example.com/templates/1</templateURL> </Template>
Parameters:	See Reply Template API Parameters, on page 143.

PUT

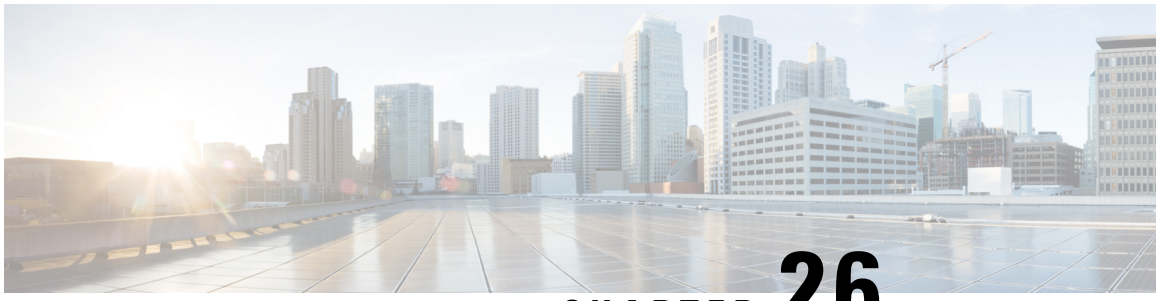
Updates an existing reply template definition.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/template/reply/<ID Variables>
HTTPS method:	PUT
Parameters:	See Reply Template API Parameters, on page 143.
Example XML request payload:	<Template> <changeStamp>0</changeStamp> <name>My Reply Template 1</name> <refURL> http://[Server]:[Port]/ccp-webapp/ccp/template/reply/100201 </refURL> <systemDefined>false</systemDefined> <templateURL>https://example.com/templates/1</templateURL> </Template>

Reply Template API Parameters

Parameters are optional unless otherwise noted.

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



CHAPTER 26

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, on page 145

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.

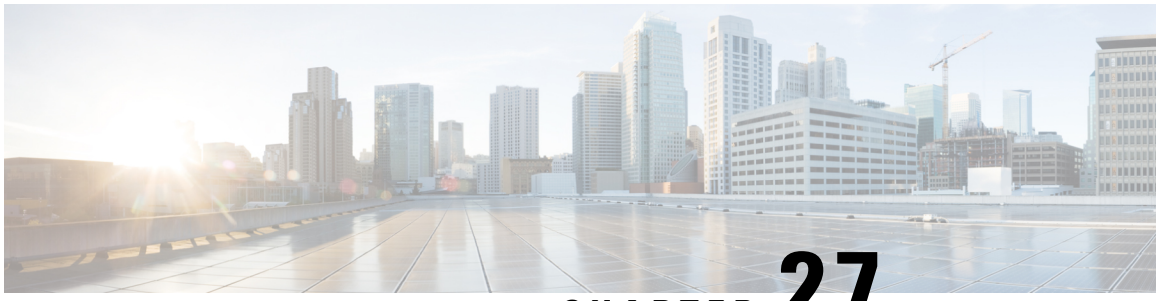
Note Only the administrator created during install can use this API.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/reportingserver
HTTPS method:	get
Example XML response:	<ReportingServer> <databaseType>Informix</databaseType> <reportingDatabase>[database name]</reportingDatabase> <reportingHost>[hostname]</reportingHost> <reportingHostIp>[ServerIP]</reportingHostIp> <reportingPort>[Port]</reportingPort> <reportingServer>[server name]</reportingServer> </ReportingServer>
Parameters	See Reporting Server API Parameters, on page 146.

Reporting Server API Parameters

All parameters are optional.

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



CHAPTER 27

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, on page 147

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 147
DELETE, on page 148
GET (List), on page 148
GET, on page 148
PUT, on page 149
Reporting User API Parameters, on page 149

POST

Creates the reporting user.

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

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/reportinguser/
HTTPS method:	POST
Parameters:	See Reporting User API Parameters, on page 149.

Example XML request payload:	<ReportingUser> <username>reportinguser</username> <password>password</password> </ReportingUser>
HTTPS response headers:	A *201 Created* http header is returned on success.

DELETE

Deletes the reporting user.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/reportinguser/<ID Variables>
HTTPS method:	DELETE
HTTPS response headers:	A *200 OK* http header is returned on success.

GET (List)

Lists all reporting users.

Note Currently only a single reporting user is supported.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/reportinguser
HTTPS method:	GET
Parameters	See Reporting User API Parameters, on page 149.
Example XML response:	<ReportingUsers> <ReportingUser> <refURL>https://[ServerIP]:[Port]/ ccp-webapp/ccp/reportinguser/100001</refURL> <username>reportinguser</username> </ReportingUser> </ReportingUsers>

GET

Gets a single reporting user from the database.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/reportinguser/<ID Variables>
HTTPS method:	GET
Example response:	<ReportingUser>

	<refURL>https://[ServerIP]:[Port]/ ccp-webapp/ccp/reportinguser/100001</refURL> <username>reportinguser</username> </ReportingUser>

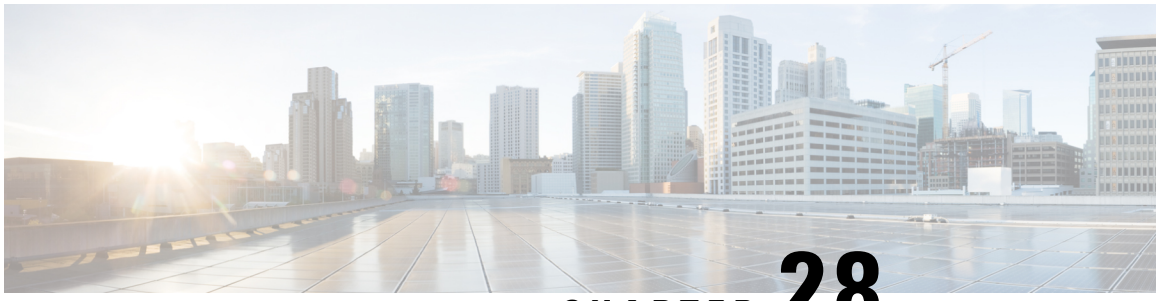
PUT

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

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/reportinguser/<ID Variables>
HTTPS method:	PUT
Parameters:	• **password**: string—the new password for the reporting user.
Example XML request payload:	<ReportingUser> <username>reportinguser</username> <password>newpassword</password> </ReportingUser>
HTTPS response headers:	A *200 OK* http header is returned on success.

Reporting User API Parameters

Parameter	Description	Notes
username	Must be *reportinguser*.	String. Required for POST
password	The password for the reporting user.	String. Required for POST



CHAPTER 28

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, on page 151

Serviceability API Commands

This section describes the supported commands for the serviceability API and the parameters for those commands. Get (Context Service Status), on page 158

Related Topics

GET

GET (List), on page 152

Get (Performance Information), on page 155

Get (Context Service Status), on page 158

Serviceability API Parameters, on page 161

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,

```
https://<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:
https://<ServerIP>:<Port>/ccp-webapp/ccp/serviceability?category=all

.

The parameter *configtype* can be used along with the category *all* or *configuration* to obtain the details of the specified configuration object along with other requested category details. Pass more than one configuration object to the *configtype* parameter by seperating the object names using a comma.

For example, the following api only returns feed configuration detail:

```
https://ServerIP:8080/ccp-webapp/ccp/serviceability?

category=configuration&configType=feed
```

Note The parameter *configtype* must be used either with the category *all* or *configuration*. Specifying an invalid value for the *configType* parameter will generate an error.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/serviceability
HTTPS method:	GET
Parameters:	See Serviceability API Parameters, on page 161.

Example XML response:	Example response for https://<ServerIP>:<Port>/ccp-webapp/ccp/serviceability/?category=serviceStates
	
	`<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>`

GET (List)

Lists all of the available serviceability attributes and their current values. For details, see the GET command for the serviceability API.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/serviceability/
HTTPS method:	GET
Example XML response:	This example is an excerpt of the XML response. The full output consists of more than 3 MB of data.

```
<Serviceability>
 <diskUsage>
  <activePartitionTotalBytes>
   13463810048
  </activePartitionTotalBytes>
  <activePartitionUsableBytes>
   2352652288
  </activePartitionUsableBytes>
  <commonDatastorePartitionTotalBytes>
   46835601408
  </commonDatastorePartitionTotalBytes>
  <commonDatastorePartitionUsableBytes>
   37941018624
  </commonDatastorePartitionUsableBytes>
  <inactivePartitionTotalBytes>
   13463781376
  </inactivePartitionTotalBytes>
  <inactivePartitionUsableBytes>
   2357100544
  </inactivePartitionUsableBytes>
 </diskUsage>
...
       <feedStatuses>
  <FeedStatus>
   <feedRefURL>
    https://[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>
...
      <chatGatewayStatus>
             <channels>
             <channel>
             <enabled>true</enabled>
             <lastFailureTimestamp>1521711982149</lastFailureTimestamp>
             <type>FACEBOOK_MESSENGER</type>
             </channel>
             </channels>
             <status>CONFIGURED</status>
       </chatGatewayStatus>
</Serviceability>
```

Get (Performance Information)

Gets information about the performance counters in the system.

URL:	https://<ServerIP>:<Port>/sm-dp/rest/DiagnosticPortal/GetPerformanceInformation

HTTPS method:	GET
Elements:	**xmlns:** XML namespace "dp" defined for Diagnostic Portal specific elements to avoid naming errors. **Schema Version:** This element is defined for tracking the version. The performance API retrieves information in the form of "Property Name" and "Value" for each of the following categories: **systemInfo:** The following counters are retrieved for the system services such as Runtime, XMPP, Webapp, Indexer, and Datastore: • System/Available Processors • System/Committed Virtual Memory Size • System/Free Physical Memory Size • System/Free Swap Space Size • System/Max File Descriptor Count • System/Open File Descriptor Count • System/Process Cpu Time • System/System Load Average • System/Total Physical Memory Size • System/Total Swap Space Size • Threading/Peak Thread Count • Threading/Thread Count • Threading/Total Started Thread Count 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 161.
Example XML response:	<dp:GetPerformanceInformationReply xmlns:dp="https://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>
```

Get (Context Service Status)

Displays the Context Service Status values.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/serviceability/contextServiceStatus
HTTPS method:	GET
Example XML response:	Example response for https://<ServerIP>:<Port>/ccp-webapp/ccp/serviceability/contextServiceStatus

```
{    "status": {
        "overallStatus": "ONLINE",
        "successfulUpgradeCount": 1,
        "failureUpgradeCount": 0
    },
    "config": {
        "appType": "socialminer",
        "orgId": "45ed7e6f-6332-470a-849d-b41892efeb16",
        "uuid": "fb40c31c-ce50-4bab-8d1f-f095b2d8c6dd",
        "staticSdkVersion": "2.0.1",
        "extensionSdkVersion": "2.0.2-10416",
        "proxy": "https://64.103.36.133:8080",
        "enabledFeatures": [{
            "name": "KMS_ENCRYPTION_KEY"
        }],
        "type": "cs_context",
        "state": "REGISTERED",
        "labMode": true,
        "requestTimeout": 5000,
        "tcpTimeout": 10000,
        "maxRetries": 1,
        "cluster": {
            "clusterId": "98c5178c-1b11-41ae-8aa3-38d140bd2c66",
            "clusterName": "socialminer-context-156807DFC8D"
        }
    },
    "services": [{
        "name": "kms",
        "url": "encryption-a.wbx2.com",
        "ping": {
            "status": "200",
            "latency": 1425
        },
        "state": "INITIALIZED"
    },
    {
        "name": "fms",
        "url": "hercules-a.wbx2.com",
```

```
        "ping": {
            "status": "200",
            "latency": 2373        }
    },
    {
        "name": "ci",
        "url": "idbroker.webex.com",
        "ping": {
            "status": "200",
            "latency": 2190        }
    },
    {
        "name": "context",
        "url": "context-service.produs1.ciscoccservice.com",
        "ping": {
            "status": "200",
            "latency": 1664
        }
    },
    {
        "name": "dictionary",
        "url": "dictionary.produs1.ciscoccservice.com",
        "ping": {
            "status": "200",
            "latency": 3412
        }
    },
    {
        "name": "ccfs",
        "url": "ccfs.ciscoccservice.com",
        "ping": {
            "status": "200",
            "latency": 1148        }
    },
    {
        "name": "discovery",
        "url": "discovery.produs1.ciscoccservice.com",
        "ping": {
            "status": "200",
            "latency": 1119
        }
    }]
}
```


HTTPS response headers:	200 OK

PUT (Triggers notification of the specified email contact in the specified campaign)

Re-notifies an already fetched email from SocialMiner back to Unified CCX.

URL:	https://<SM_FQDN>/ccp-webapp/ccp/campaign/{publicid_of_campaign}/notification/contact/{contactid
HTTPS method:	PUT
Parameters	None

HTTPS Response Headers:	200 OK 401 UNAUTHORIZED 403 FORBIDDEN 500 INTERNAL SERVER ERROR

POST

Creates folder configuration in the email server before configuring the email feed.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/emailConfigTest
HTTPS method:	POST
Input/Output method:	xml, json.
Parameters:	See Serviceability API Parameters, on page 161.
Example XML request payload:	<EmailServerDetails> <server> <!-- SMTP Server Details --> <smtp> <host>smtp.office365.com</host> <port>587</port> </smtp> <!-- IMAP server details --> <imap> <host>outlook.office365.com</host> <port>993</port> <!-- IMAP folder List --> <folders> <folder>Inbox</folder> <folder>Myfolder</folder> </folders> </imap> </server> <!-- Server Credentials --> <userName>name@example.com</userName> <password>1234</password> </EmailServerDetails>
Example Response Body:	<EmailServerTestResult> <!-- Overall Status SUCCESS/FAILURE--> <globalStatus>FAILURE</globalStatus> <smtp> <!-- status SUCCESS/FAILURE --> <status>SUCCESS</status> <!-- Failure reason --> <reason></reason> </smtp> <imap> <!-- IMAP folder Accessibility results --> <folders> <folder> <folderName>Inbox</folderName> <status>SUCCESS</status> <reason></reason>

```
      </folder>
      <folder>
        <folderName>Myfolder</folderName>
        <status>FAILURE</status>
        <reason>FOLDER MISSING</reason>
      </folder>
    </folders>
   </imap>
</EmailServerTestResult>
```

Serviceability API Parameters

Parameters are optional unless otherwise noted.

Parameter	Description	Notes
chatGatewayStatus	The current status of the SocialMiner Chat Gateway that is used for integration with Facebook Messenger and any other messaging service.	• status - one of the following: • CONFIGURED: valid configuration with at least one channel • NOT_CONFIGURED: service has not been configured with any channels • ERROR: it is possible that the service failed to read configuration and is in an ERRONEOUS state, or the service itself is DOWN or UNREACHABLE • channels - A list of channels that are currently configured, and their status (This will be missing if the **status** is **NOT_CONFIGURED** or **ERROR**) • channel - The status of an individual configured channel • type - The type of this channel. Example: **FACEBOOK_MESSENGER** • enabled - will be **true** if the channel is configured and enabled • lastFailureTimestamp - An epoch value containing the timestamp of the last failure that occurred in this channel. Useful for tracking any intermittent failures etc. This is reset each time the configuration for that channel changes.

Parameter	Description	Notes
configuration	An XML representation of all configuration objects in the system.	This is the same XML that the API produces and consumes.
diskusage	Reports on the total size in bytes and remaining usable bytes for each partition on the disk.	In some small deployments, there will be no reportingDatabase partition because it is the same as the commonDatastore partition. In large deployments, the partitions reported for both total and usable size will be: • activePartition—main OS partition. Contains config database. Mounted on */*. • inactivePartition—inactive side used to upgrade. Will be activePartition after a switch. Mounted on */partB*. • commonDatastorePartition—social contact datastore. Both Cassandra and Solr data files. Mounted on */common*. • reportingDatabasePartition—contains the reporting database. Mounted on */spare*. On small deployments this is sometimes a symlink into /common. In those cases, this partition will be omitted from the results.

Parameter	Description	Notes

eventingInfo	Statistics about the Eventing Subsystem which publishes social contact state change events to XMPP clients using the embedded XMPP server.	• dsNfyMsgsRcvd: the number of social contact state change messages received from the data store.
		• xmppEventsSent: the number of xmpp events successfully sent by this notifier.
		• xmppEventFailed: the number of failed xmpp notification attempts.
		• xmppEventDropped: the number of xmpp events that were dropped because the output queue was full.
		• outQueueDepth: the number of items in the output queue.
		• outQueueWait: the average amount of time between when an xmpp event is queued and when it is sent.
		• connectionStatus: one of the following states:
		  • *CONNECTED*: successfully connected to the embedded XMPP server.
		  • *DISCONNECTED*: the notifier is unable to connect to the configured server.
		  • *DISABLED*: the notifier is not enabled.
		  • *BAD_CONFIGURATION*: the notifier is not configured correctly and is unable to attempt to make a connection.

feedstatuses	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. See Feed statusDescription Values, on page 172.	FeedStatus can include this information: • FETCH_SUCCESS_RATE: calculated based on the last 10 fetches: FetchSuccessRate = ((FetchSuccess / (FetchSuccess+FetchFailure))/100). Applies to these classes: ccp-feeds and ccp-webapp. • LAST_GOOD_FETCH_TIME: applies to classes ccp-feeds and ccp-webapp.

Parameter	Description	Notes
filterStatuses	FilterStatus elements containing the last known status of every filter in the system.	• name: filter name • status: 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. • status description: a detailed error message in the case of a script filter execution error. Otherwise, normal. • timeSinceLastExecution: 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.
hardware	The status of the hardware SMALL, LARGE, UNSUPPORTED.	diskOneSize: size of disk 1. diskTwoSize: size of disk 2. diskOneStar: start byte of disk 1. diskTwoStart: start byte of disk 2. memory: total physical memory. numberOfCpus: total number of cpus(cores). 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.	The stats for each JVM are: • connectionName: the name and JMX URL (only accessible from the local machine) of the JVM. • cpuPercentageSamples: samples of the percentage of CPU used by this JVM. Each sample is a 30 second interval. Five days of samples are kept. • heapSamples: samples of the total heap usage in MB. Each sample is a 2 minute interval. One week of samples are kept. • heapSlope: 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. • uptime: elapsed time since this JVM started in milliseconds. • uptimeString: a human readable version of uptime.

Parameter	Description	Notes
notifiers	The status of the notification services for this server.	**For "email", "im" and "http" notification types, associated data are** • type: the type of notification sent by this notifier. • notificationsSent: the number of notifications successfully sent by this notifier. • notificationsFailed: the number of failed notifications. • notificationsDropped: the number of notifications that were dropped because the output queue was full. • outQueueDepth: the number of items in the output queue. • outQueueWait: the average amount of time in milliseconds between when a notification request is queued and when it is sent.
		• **ConnectionStatus**:Can be one of • *CONNECTED*: the notifier successfully connected to the configured server. • *DISCONNECTED*: the notifier is unable to connect to the configured server. • *DISABLED*: the notifier is not enabled. • *BAD_CONFIGURATION*: the notifier is not configured correctly and is unable to attempt to make a connection.

Parameter	Description	Notes
notifiers		**For "connection to cce" notification type, associated data are** • type: connection to cce. • notificationsSent: the number of tasks successfully allocated to an agent by cce. • notificationsFailed: the number of tasks failed to find an agent. • notificationsDropped: the number of tasks that were dropped because there was no connection to cce. • outQueueDepth: the number of tasks currently being processed by cce. • outQueueWait: 0.
		• **ConnectionStatus**: Can be one of • *NOT_ESTABLISHED*: CCE configuration for multichannel routing is disabled. • *LISTENING*: listening for incoming connection from CCE media routing PG. • *ESTABLISHING*: establishing connection from CCE media routing PG. • *ESTABLISHED_ ROUTING_ ENABLED*: established connection with CCE media routing PG.

Parameter	Description	Notes

notificationRule Statuses	Status and statistics for each notification rule.	• id: the id of the notification rule.
		• name: the name of the notification rule.
		• type: the type of notification rule.
		• status: an explanation of the status for the most recent notification sent.
		  • success: the number of notifications that were successfully sent.
		  • failure: the number of notifications that failed to be sent.
		  • total: The total number of notifications processed.

Parameter	Description	Notes
		• statusReason: Reason code for the current status. • **NOTIFICATION_ STATUS_ UNEXPECTED_ ERROR** • **NOTIFICATION_ STATUS_ NORMAL** • **NOTIFICATION_ STATUS_BAD_ CONFIGURATION** • **NOTIFICATION_ STATUS_ CONNECTION_ PROBLEM** • **NOTIFICATION_ STATUS_ AUTHENTICATION_ FAILED** • **NOTIFICATION_ STATUS_CCE_ ROUTING_PROBLEM** • **NOTIFICATION_ STATUS_ NO_CONNECTION_ TO_CCE** • **NOTIFICATION_ STATUS_ CCE_MESSAGE_ QUEUE_FULL** • **NOTIFICATION_ STATUS_**

Parameter	Description	Notes
		RATE_LIMITED
		• statusReasonDescription: an explanation for the current statusReason code. • statusChangeTime: time when the status was last changed.
serviceStates	Get the state of services.	Values for serverState can be: • SERVER_STATE_API_INIT: the Serviceability API is initializing. This is the state when the API is first started. • 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. • SERVER_STATE_IN_SERVICE: the Runtime, Datastore, or Indexer server is in service. • 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.
sessionInfo	A list of current sessions (ip address:username combination.	This is used to indicate active sessions and calculate approximate number of logged in users.

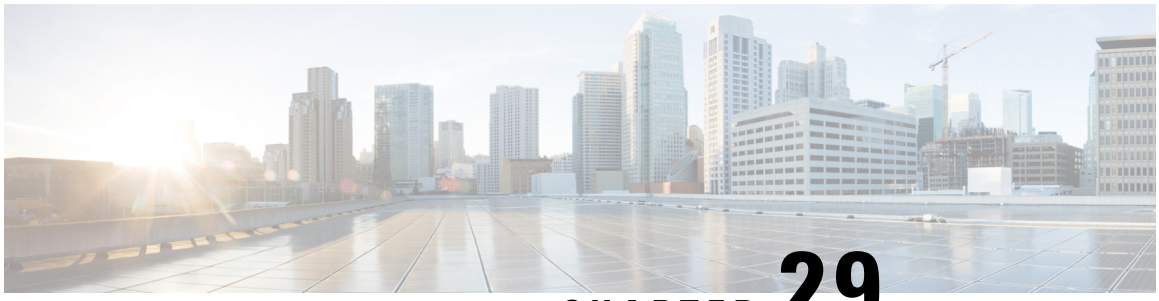
Parameter	Description	Notes
subsystems	Get critical subsystem parameters.	This category reports the same type of information as the systemInfo category, but reports on only the following critical properties: • PeakThreadCount • TotalStartedThreadCount • ThreadCount • CommittedVirtualMemorySize • FreePhysicalMemorySize • FreeSwapSpaceSize • MaxFileDescriptorCount • OpenFileDescriptorCount • ProcessCpuTime • SystemLoadAverage • TotalPhysicalMemorySize • TotalSwapSpaceSize Accessing the subsystems category has less impact on system performance than accessing the systemInfo category.
systemInfo	Get all available system parameters.	**Important!** 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.
systemConditions	Get a list of SystemCondition elements describing persistent states reported by the services.	
systemPerformance	**socialContactsPerHour:** 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.
version	Application version information.	Shows the currently installed version of SocialMiner and the prior version of SocialMiner.
host	The URL of the email server.	

Parameter	Description	Notes
port	The port used by the email server.	
folder	The name of the folder in the email server.	
username	The email address used to connect to the email server.	
password	The password used to connect to the email server.	

Feed statusDescription Values

A feed statusDescription can have the following values:

- **AUTHENTICATION_FAILED**: An authenticated feed failed because of incorrect credentials.
- **DATASTORE_ERROR**: Error attempting to write the social contacts to the datastore.
- **EMAIL_CONNECT_ERROR**: Cannot establish a connection with the email server. Check that the email server, username, and password are configured correctly.
- **EMAIL_FOLDER_DOES_NOT_EXIST**: The selected email folder cannot be found. Check that the folder is correct and exists in the user's mailbox.
- **EMAIL_IMAP_NO_CLIENT**: The email client is missing.
- **EMAIL_MESSAGING_EXCEPTION**: An error occurred while retrieving email from the email server.
- **EMAIL_NO_SUCH_PROVIDER**: The configured email protocol is not supported.
- **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.
- **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.
- **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.



CHAPTER 29

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, on page 173

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 https 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.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/socialcontact/
HTTPS method:	POST
Parameters:	See Social Contact API Parameters, on page 183.
Example XML request payload:	<SocialContact> <feedRefURL>https://[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.

HTTPS 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="https://<ServerIP>:<Port>/ccp-webapp/ccp/socialcontact/22E00F5310000129460A1EB40A568DDE" />

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/socialcontact/<ID Variables>
HTTPS method:	GET

Example responses:	Results are returned as XML.
	`<SocialContact>`
	`  <author>Michael Littlefoot</author>`
	`  <createdDate>1531136305099</createdDate>`
	`  <description>some description</description>`

```
    <draftAction>draft_action_reply</draftAction>
    <draftResponse/>
    <extensionFields>
        <extensionField>
            <name>h_chatRoom</name>
            <value>socialminer_chat.0@conference.127.0.0.1</value>
        </extensionField>
        <extensionField>
            <name>h_Name</name>
            <value>Michael Littlefoot</value>
        </extensionField>
    </extensionFields>
    <id>7ED5E7C5100001640000000000A4E5F26</id>
    <inviteStatus>none</inviteStatus>
    <isInvited>false</isInvited>
    <isSoftLocked>false</isSoftLocked>
    <link>
        https://[Server]:[Port]/ccp-webapp/ccp/socialcontact
       /7ED5E7C5100001640000000000A4E5F26
    </link>
    <publishedDate>1531136305092</publishedDate>
    <refURL>
        https://[Server]:[Port]/ccp-webapp/ccp/socialcontact
       /7ED5E7C5100001640000000000A4E5F26
    </refURL>
    <replyTemplateRefURL>
        https://[Server]:[Port]/ccp-webapp/ccp/template/reply/303
    </replyTemplateRefURL>
    <replyTemplateURL>

https://[Server]:[Port]/multisession/ui/templates/reply/cisco_agent_ms_chat.jsp

    </replyTemplateURL>
    <replyToId/>
    <shortUrlIds/>
    <sourceType>chat</sourceType>
    <status>unread</status>
    <statusReason>unknown</statusReason>
    <statusTimestamp>1531136306619</statusTimestamp>
    <statusUserId>admin</statusUserId>
    <tags>
        <tag>tag1</tag>
        <tag>tag2</tag>
    </tags>
    <title>Chat from Website</title>
    <transcriptRefURL>
        https://[Server]:[Port]/ccp-webapp/ccp/socialcontact/
        7ED5E7C5100001640000000000A4E5F26/transcript
    </transcriptRefURL>
</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.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/socialcontact/<ID Variables>
HTTPS method:	PUT
Parameters:	See Social Contact API Parameters, on page 183.
Example XML request payload:	<SocialContact> <status>reserved</status> <statusReason>unknown</statusReason> <statusTimestamp>1531136406619</statusTimestamp> <tags> <tag>changed_1</tag> <tag>changed_2</tag> </tags> </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>https://[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>

PUT (Requeue Email)

Update an existing email contact for requeue.

Agents use the requeue feature to requeue email contacts to a different Contact Service Queue (CSQ). When email contacts are requeued, the from address in the reply must be set to the address that corresponds to the feed associated with the CSQ to which the contact is requeued. For more information, refer to your Unified Contact Center Express documentation.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/socialcontact/<id>/requeue
HTTPS method:	PUT
Input/Output format:	XML
Parameters:	• *statusTimestamp* (required). The timestamp of the last change of the social contact, which is returned in a GET request. • *status* (optional). The status to set for the email contact. SocialMiner only accepts a status of RESERVED (combined with a statusReason that indicates that the contact is being rerouted) and rejects any other status. The status parameter is case-sensitive. • *extensionFields* The following extensionField is required • feedTag. Provide a key-value pair with the key *feedTag* and a value that indicates the unique tag configured for the feed that corresponds to the CSQ to which the agent or Unified CCX wants to requeue the email contact. • *statusReason* (required, case-insensitive). The reason that the contact is requeued. Valid values are as follows: • EMAIL_REQUEUE_TRANSFER. This reason is used when the client requeues the contact. • EMAIL_REQUEUE_AGENT_DISCONNECTED. This reason is used to requeue the contact when an agent is disconnected.

	Note When the state of an email contact is RESERVED and the statusReason is a requeue reason, this API rejects any new requeue request for the same contact. After the rerouted contact is accepted by an agent and the statusReason is updated to a reason that is not a requeue reason, the contact can be requeued. This API uses the feedTag parameter to find the feed with that tag in the database. Each email feed has only one unique tag. The API then populates the ID of the feed into the requeueFeedId extension field. The reply template uses the requeueFeedId to look up the from address to use to reply to the customer. If the API does not find a feed with the indicated feedTag, the requeue proceeds without populating or overwriting the requeueFeedId extension field. Therefore, the reply template does not have the new from address for the reply. The reply contains the last valid from address. If more than one feed exists with the tag indicated in the feedTag parameter, the API populates or overwrites the requeueFeedId with the first match that it finds. The API does not delete any existing extension fields. The API preserves existing extension fields and merges in any newly-provided extension fields (with the exception of the requeueFeedId extension field as described in the previous cases).

Example XML request payload :	<pre><SocialContact>
	  <extensionFields>
	  <extensionField>
	  <name>feedTag</name>
	  <value>email_csq3</value>
	  </extensionField>
	  </extensionFields>
	  <statusReason>email_requeue_transfer</statusReason>
	  <statusTimestamp>140580003345</statusTimestamp>
	</SocialContact></pre>

HTTPS response headers:	A 200 OK header is returned on success. A 4xx Bad Response is returned if • The contact does not exist. • The contact is not of sourceType email. • A statusTimestamp mismatch exists. • The provided status is not RESERVED. • The current status of the contact is not RESERVED. • The statusReason is not valid. • The current statusReason is already a requeue reason. • The required extension field (feedTag) is missing. **Note** For multisession contacts, the status update uses the scId of the contact as its statusUserId.

Example XML response payload :	<SocialContact> <extensionFields> <extensionField> <name>emailUniqueId</name> <value>101</value> </extensionField> <extensionField> <name>emailReplyTo</name> <value>Breakfast Club < bclub@email13.sm ></value> </extensionField> <extensionField> <name>feedTag</name> <value>email_csq3</value> </extensionField> <extensionField> <name>requeueFeedId</name> <value>10001</value> </extensionField> </extensionFields> <refURL> https://[Server]:[Port]/ccp-webapp/ccp/socialcontact/ 91C52DD6100001470000001940A56866</refURL> <status>reserved</status> <statusReason>email_requeue_transfer</statusReason> <statusTimestamp>1407600001015</statusTimestamp> <statusUserId>91c52dd6100001470000001940a568665</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.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/socialcontact/<ID Variables>/transcript
HTTPS method:	GET
Parameters:	See Social Contact API Parameters, on page 183.
Example XML response :	<ChatTranscript> <id>4C01DF3B10000145000010F60A568DD9</id> <refURL>https://[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/ 4C01DF3B10000145000010F60A568DD9/transcript</refURL> <endDate>1326918175931</endDate> <participants> <participant> <agentName/> <nickName>Tetyana</nickName> </participant> <participant> <agentName>admin</agentName> <nickName>Agent</nickName> </participant> </participants> <startDate>1326918153649</startDate>

```
    <transcript>
        <chat>
            <time>1326918172743</time>
            <name>Agent</name>
            <msg>hello</msg>
        </chat>
        <chat>
            <time>1326918173170</time>
            <name>Agent</name>
            <msg>How can I help you?</msg>
        </chat>
    </transcript>
</ChatTranscript>
```

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:

https://<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).
- 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 https://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.

URL:	https://ServerIP:8080/ccp-webapp/ccp/search/contacts

HTTPS method:	GET
Output format:	ATOM
Parameters:	• *q* (required). Query parameter. • *count* (optional.) Defines how many results to return (default = 50, max = 200). • *startIndex* (optional). Identifies the index of the first search result. Because we are using Solr for querying, the first index should be 0 (default = 0).
Example query request:	`https://<ServerIP>:<Port>/ccp-webapp/ccp/search/contacts?q=chat.agentName:admin%20AND%20sc.socialContactStatus:handled` 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").
Example XML response:	<?xml version="1.0" encoding="UTF-8"?> <feed xmlns="http://www.w3.org/2005/Atom" xmlns:dc= "http://purl.org/dc/elements/1.1/" xmlns:ccp="https://www.cisco.com/ccbu/ccp/xml/socialcontact/1.0/" xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/"> <title>Results of Search</title> <link rel="self" href=" https://<ServerIP>:<Port>/ccp-webapp/ccp/ search/contacts?q=chat.agentName:admin%20AND%20sc.socialContactStatus: handled&count=50&startIndex=0" /> <subtitle>This feed has been created by Cisco SocialMiner</subtitle> <id>https://<ServerIP>:<Port>/ccp-webapp/ccp/search/contacts</id> <updated>2014-01-14T18:46:27Z</updated> <dc:date>2014-01-14T18:46:27Z</dc:date> <opensearch:itemsPerPage>1</opensearch:itemsPerPage> <opensearch:totalResults>1</opensearch:totalResults> <opensearch:startIndex>1</opensearch:startIndex> <opensearch:Query role="request" searchTerms="chat.nickname:Tetyana"/> <opensearch:link rel="search" type="application/opensearchdescription+xml" href="https://www.cisco.com/opensearch-description.xml" /> <entry> <title>title14</title> <link rel="alternate" href="https://<ServerIP>:<Port>/ccp-webapp/ccp/ socialcontact/91E6B21910000143000032B10A568DD9" /> <link rel="socialcontact" href="https://<ServerIP>:<Port>/ccp-webapp/ccp/ socialcontact/91E6B21910000143000032B10A568DD9" /> <author> <name>author14</name> </author> <id>https://<ServerIP>:<Port>/ccp/socialcontact/ 91E6B21910000143000032B10A568DD9</id> <updated>2014-01-14T17:57:28Z</updated> <published>2014-01-14T17:57:28Z</published> <content type="application/xml"> <ChatTranscript xmlns=""> <id>F27C35DD10000134000007160A568DDD</id> <endDate>1326918175931</endDate> <participants> <participant> <agentName>admin</agentName>

```
              <nickName>Agent</nickName>
            </participant>
          </participants>
          <startDate>1326918153649</startDate>
          <transcript>
            <chat>
              <time>1326918172743</time>
              <name>Agent</name>
              <msg>Hello, how can I help you?</msg>
            </chat>
            <chat>
              <time>1326918173170</time>
              <name>author14</name>
              <msg>Never mind. I'm all set, thanks.</msg>
            </chat>
          </transcript>
        </ChatTranscript>
      </content>
      <dc:creator>author14</dc:creator>
      <dc:date>2012-01-18T20:22:33Z</dc:date>
      <ccp:scstatustimestamp>1326918177731</ccp:scstatustimestamp>
      <ccp:scstatus>handled</ccp:scstatus>
      <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
author	The social contact author name.	String. Required for POST.
description	The body of the social contact.	String.

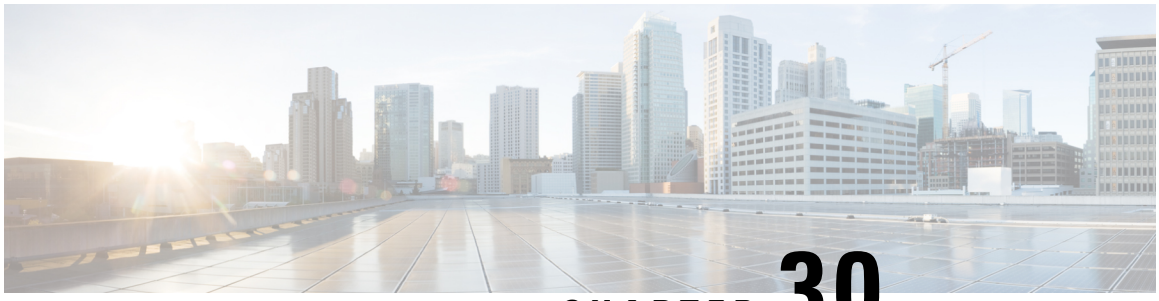
Parameter	Description	Notes
extensionField	A wrapper tag for a custom name and value pair.	For email contacts, SocialMiner stores the following attributes in the extension fields: • *emailUniqueIdInFolder*: The ID of the email message. The ID identifies the email message in its folder. The ID is not valid if the email is moved from the folder associated with the feed that captured the email. • *emailFrom*: A comma-separated list of email users who authored the email. • *emailReplyTo*: A comma-separated list of the email users to include in a reply to the email message. • *agentId*: The agent who is currently handling the email contact. If an agent replies to a customer email and the customer replies to the agent's reply, this field can be used to route the customer reply to the same agent. • *customer_attachment_names*: A quoted and comma-separated list of email attachment filenames Quotes in filenames are escaped using a backslash quote (/") For example: "file1.txt","file2.txt"

Parameter	Description	Notes
extensionFields	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. 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.
feedRefURL	The feed refURL that the social contact is associated with.	String. Required for POST.
isSoftLocked	Indicates that the social contact should not be modified, but is not enforced in the API.	Boolean. 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.
isInvited	Indicates if the social contact was created as a result of a chat invitation.	Boolean. Default is false. If the Social Contact was created as a result of a chat invitation, then isInvited must be set to true.
inviteStatus	The status of chat invitations sent (if any) from this social contact.	String, case-insensitive. Default is NONE. Valid values are: • NONE (no chat invitations were sent from this social contact) • SENT (chat invitations were sent from this social contact) • EXPIRED (chat invitations were sent but were not accepted by the customer)

Parameter	Description	Notes
link	References the original source of the social contact. For email messages, SocialMiner creates a link in the following format: `https://<emailServerAddress>?u=<toEmailUsername>&f=<emailFolder>&m=<emailUniqueIdInFolder>&t=<emailReceivedTimestamp>`	The link for email contacts includes the following information: • *emailServerAddress*: The email server that retrieved the contact (the email receive host associated with the feed that retrieved the contact) • *toEmailUsername*: The email user who received the email (the email username associated with the feed that retrieved the contact) • *emailFolder*: The folder that holds the email (the email.receive.folderName associated with the feed that retrieved the contact) • *emailUniqueIdInFolder*: The unique ID that identifies the message in this folder • *emailReceivedTimestamp*: The time that the email server received the message
publishedDate	The social contact published date. For contacts with sourceType email, this parameter is the date that the email server received the email message.	String. Leave blank to use the current timestamp or provide a valid Unix timestamp.
refURL	A copy of the URL requested.	
replyTemplateRefURL	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.
replyTemplateURL	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.

Parameter	Description	Notes
shortUrlId/shortUrlIds	Is a list of short URL Ids which were generated for this social contact.	String. 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.
sourceType	Is the type of feed this social contact came from: callback, chat, or push.	String. This is set by SocialMiner and cannot be set through the create or update APIs.
status	One of: • **unread**—The default state of a new contact. • **reserved**—Reserved to be handled. • **handled**—This contact has been handled and no further action is required. • **discarded**—This contact does not require a response and is filed in the recycle bin. • **queued**—The contact is in the process of or has been routed to some external entity. • **draft**—A draft response to the contact has been created and saved, but not sent.	String (case-sensitive). 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. 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.
statusReason	The reason why the contact is in the current state.	
statusUserId	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
statusTimestamp	The time stamp of the last state change of the social contact.	Long integer. Required for PUT. **Important**: 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 **get** request, then the update fails. This mechanism is in place so that two clients cannot update the same social contact at the same time. The statusTimestamp changes to the current timestamp if the update is successful.
tag/tags	One or more tags to associate with this social contact.	The tags can be new or existing tags. If you include the *tags* element, but do not include any *tag* elements, then tags are deleted during an update.
title	The title of the social contact.	String. Required for POST.
transcriptRefURL	Is a URL to get the chat transcript.	String. Only applies to contacts where sourceType is chat.



CHAPTER 30

Socks Proxy

The Socks Proxy Configuration API allows you to configure proxy details used by SocialMiner to communicate with cloud based email services like Office 365.

- Socks Proxy API Commands, on page 189

Socks Proxy API Commands

This section describes the supported commands for the configuration of Socks Proxy for Cloud-based Email services.

Related Topics

GET, on page 189

PUT, on page 190

Socks Proxy API Parameters, on page 190

GET

Retrieves the socks proxy configuration.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/socksProxy https://<ServerIP>:<Port>/ccp-webapp/ccp/socksProxy
HTTPS method:	GET
Parameters:	See Socks Proxy API Parameters, on page 190.

Example XML request payload:	`<SocksProxy>`
	`  <enabled>true</enabled>`
	`  <hostname>socksserver IP or FQDN</hostname>`
	`  <port>1080</port>`
	`  <username>username</username>`
	`  <password>*****</password>`
	`</SocksProxy>`

HTTPS response headers:	401 Unauthorized 500 Internal Server Error

	200 OK: Success

PUT

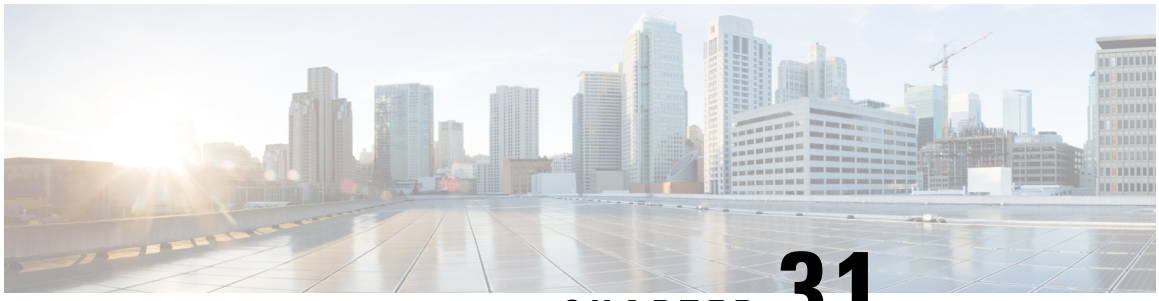
Updates the socks proxy configuration.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/socksProxy https://<ServerIP>:<Port>/ccp-webapp/ccp/socksProxy
HTTPS method:	PUT
Parameters:	See Socks Proxy API Parameters, on page 190
Example XML request payload:	<SocksProxy> <enabled>true</enabled> <hostname>socksserver IP or FQDN</hostname> <port>1080</port> <username>username</username> <password>password</password> </SocksProxy>
HTTPS response headers:	400 Bad Request 401 Unauthorized 500 Internal Server Error 200 OK: Success

Socks Proxy API Parameters

Parameters are optional unless otherwise noted.

Parameter	Description	Notes
enabled	Whether Socks Proxy is enabled.	Boolean. Allowable values are True or False.
hostname	The IP Address or the hostname of the Socks5 server.	String. This is a mandatory parameter and should be a valid FQDN/IP Address.
port	The port used by the Socks5 server.	Integer. This is a mandatory parameter and should be a valid 16 bit unix port.
username	Username to connect to the Socks5 server.	String.
password	Password to connect to the Socks5 server.	String. If username is specified, it is mandatory to specify the password.



CHAPTER 31

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 173).

- Tag API Command, on page 191

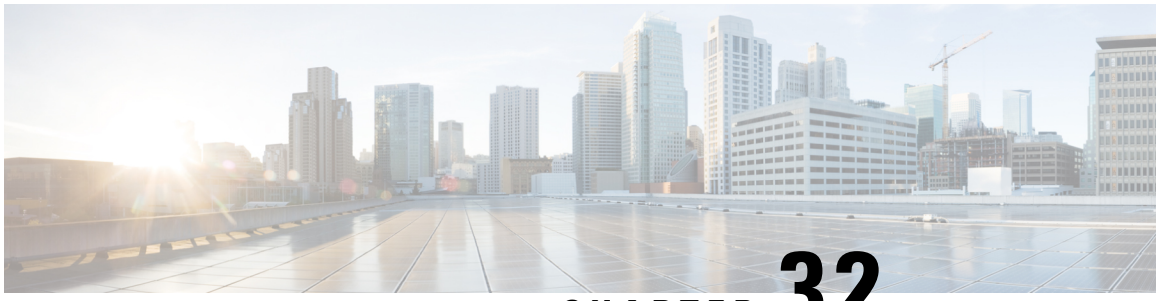
Tag API Command

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:	https://<ServerIP>:<Port>/ccp-webapp/ccp/tag
HTTPS method:	GET
Example XML response:	<Tags> <Tag> <name>tagname1</name> </Tag> <Tag> <name>tagname2</name> </Tag> ... </Tags>



CHAPTER 32

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, on page 193

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.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/shorturl/
HTTPS method:	POST
Parameters:	**Type**: optional. The shortURL type. Valid values are generic (the default) and chat_invite. **URL**: required. The URL being shortened (required). **campaignRefURL**: required for chat_invite type. The refURL of the campaign for which this short URL is being created. **scRefURL**: required for chat_invite type. The refURL of the inviting social contact.

	active: 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. **expireDate**: 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.
Example XML request payload:	<ShortURL> <type>chat_invite</type> <url>http://theurl.com?param1=val1</url> <campaignRefURL>https://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/[public ID]</campaignRefURL> <scRefURL>https://[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/[SC ID]</scRefURL> <expireDate>[timestamp]</expireDate> </ShortURL> Or, for a generic shortURL: <ShortURL> <url>http://theurl.com?param1=val1</url> </ShortURL> **Note** When using a full url with multiple parameters, special characters must be properly escaped. This means, for example, that characters such as "&" should appear inside the xml body as "&".
HTTPS response headers:	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. See API Conventions for error information.

GET

Get a shortened URL.

URL:	https:// <ServerIP>:<Port> /ccp-webapp/ccp/shorturl/<id>
HTTPS method:	GET
Example XML response payload:	<ShortURL> <type>chat_invite</type> <active>false</active> <url>http://theurl.com?param1=val1</url> <creator>[userID]</creator> <campaignRefURL>https://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/[public ID]</campaignRefURL> <scRefURL>https://[ServerIP]:[Port]/ccp/socialcontact/[SC ID]</scRefURL> <createdDate>[timestamp]</createdDate> <expireDate>[timestamp]</expireDate> <usedDate>[timestamp]</usedDate> <shortURL>/ccp/s/[id]</shortURL>

	<refURL>https://[ServerIP]:[Port]/ccp-webapp/ccp/shorturl/[id]</refURL> </ShortURL> The "refURL" is a copy of the URL requested.
Elements:	**type**: the shortURL type. **URL**: the URL being shortened. **creator**: user that created the short URL. **active**: for the chat_invite type, active means that the shortURL was successfully sent with the reply as a chat invitation. **campaignRefURL**: the refURL of the campaign for which this short URL was created. **scRefURL**: the refURL of the inviting social contact. **createdDate**: timestamp on which short URL was created. **expireDate**: the expiration date of the shortened URL. **usedDate**: the timestamp on which this URL was used. Empty if not used yet. **shortURL**: the shortened URL absolute path on the SocialMiner server. **refURL**: the URL of this short URL object.
HTTPS response headers:	See HTTPS Responses.

GET (List)

List all shortened URLs.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/shorturl/
HTTPS method:	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>https://[ServerIP]:[Port] /ccp-webapp/ccp/shorturl/[id]</refURL> </ShortURL> <ShortURL> <type>chat_invite</type> <active>true</active> <url>https://cisco.com/index.htm</url>

	<creator>[userID]</creator> <campaignRefURL>https://[ServerIP]:[Port]/ccp-webapp/ccp/campaign/[public ID]</campaignRefURL> <scRefURL>https://[ServerIP]:[Port]/ccp-webapp/ccp/socialcontact/[SC ID]</scRefURL> <expireDate>[timestamp]</expireDate> <usedDate>[timestamp]</usedDate> <shortURL>/ccp-webapp/ccp/s/[id]</shortURL> <refURL>https://[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.

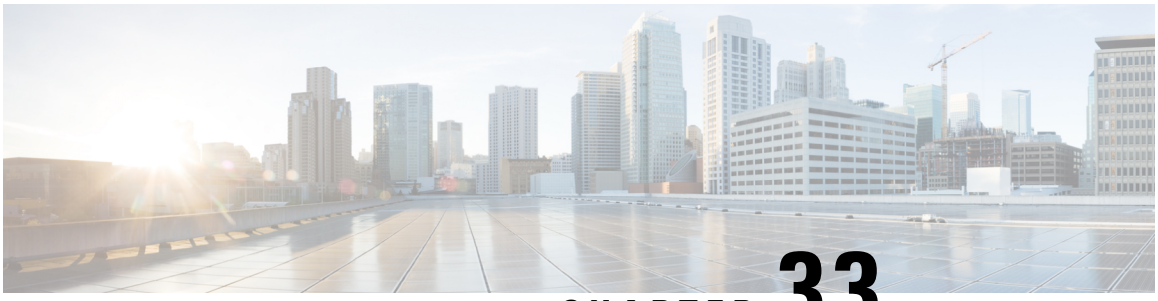
Any attempt to update other fields will result in an error.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/shorturl/<id>
HTTPS method:	PUT
Example XML request payload:	<ShortURL> <active>true</active> <usedDate>[timestamp]</usedDate> </ShortURL>
HTTPS response headers:	See HTTPS Responses.

DELETE

Delete a shortened URL.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/shorturl/<id>
HTTPS method:	DELETE
HTTPS response headers:	See API Conventions for error information.



CHAPTER 33

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, on page 197

XMPP API Commands

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

Related Topics

GET, on page 197
PUT, on page 198
XMPP API Parameters, on page 198

GET

Get the XMPP configuration.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/xmpp/default
HTTPS method:	GET
Example XML output:	<Xmpp> <xmppService>xmpp.cisco.com</xmppService> <xmppHost>[ServerIP]</xmppHost> <xmppPort>[Port]</xmppPort> <xmppServiceLookup>true</xmppServiceLookup> <xmppEnabled>true</xmppEnabled> <xmppServiceUsername>gding</xmppServiceUsername> <xmppServicePassword>...</xmppServicePassword> <refURL>

	<pre>https://[ServerIP]:[Port]/ccp-webapp/ccp/
	xmpp/default
	</refURL>
	</Xmpp></pre>

Parameters:	See XMPP API Parameters, on page 198.

PUT

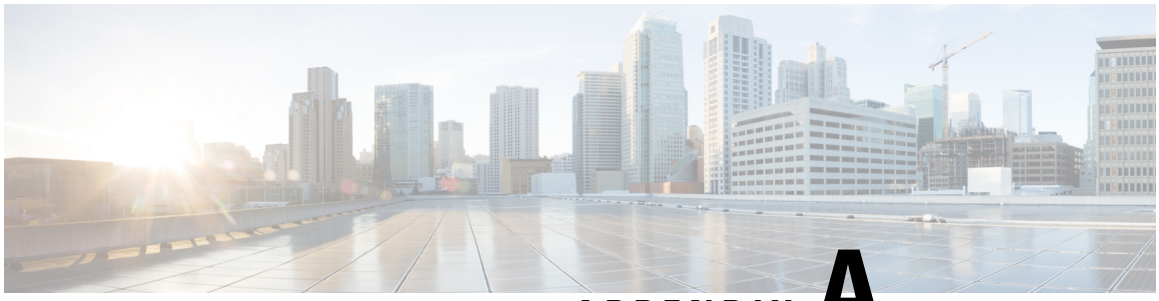
Updates the XMPP configuration.

URL:	https://<ServerIP>:<Port>/ccp-webapp/ccp/xmpp/default
HTTPS method:	PUT
Parameters:	See XMPP API Parameters, on page 198.

XMPP API Parameters

All parameters are optional.

Parameter	Description	Notes
xmppEnabled	Identifies whether this XMPP configuration is enabled or disabled.	Boolean. Default is false.
xmppHost	The IP address or hostname of the XMPP server.	String.
xmppPort	The XMPP port number. The default port is 5222.	Integer. Not used if *xmppServiceLookup* is set to *true*.
xmppService	The xmpp service lookup name.	String.
xmppServiceLookup	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.
xmppServiceUserName	The username used to log into the XMPP server.	String.
xmppServicePassword	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, on page 199
- Reporting Database Schema, on page 199

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:
campaignid	Auto-incrementing surrogate ID	serial (8)	Primary key, not Null.
configcampaignid	The internal database ID.	int (8)	Not Null.
campaignname	The campaign name, as defined in the campaign panel.	nvarchar	Not Null.
lastupdated	Last time this row was updated.	datetime	Not Null.
active	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:
recordid	Auto-incrementing ID	serial(8)	Primary key, not Null.
interval	Reporting interval to which this record applies (currently only a 15 minute interval is supported).	datetime	Not Null.
campaignid	ID of the campaign from the mmca_reportcampaign table.	int(8)	Not Null.
screceived	The number of social contacts that were received for this campaign for this interval.	int(8)	Not Null.
screserved	The number of social contacts that were reserved for this campaign for this interval.	int	Not Null.

Field name:	Description:	Data type:	Constraints:
schandled	The number of social contacts that were handled for this campaign for this interval.	int	Not Null.
scdiscarded	The number of social contacts that were discarded for this campaign for this interval.	int	Not Null.
reservedtime	Cumulative reserved time for all social contacts reserved in this campaign for this interval. Reserved time is the time between when the contact was received and when the contact was marked as reserved.	bigint	Not Null.
handledtime	Cumulative handled time for all social contacts handled in this campaign for this interval. Handled time is the time between when the contact was received and when the contact was marked as handled.	bigint	Not Null
discardedtime	Cumulative discard time for all social contacts discarded in this campaign for this interval. Discard time is the time between when the contact was received and when the contact was marked as discarded.	bigint	Not Null
chatinvitationssent	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

Field name:	Description:	Data type:	Constraints:
chatinvitationshandled	The number of chat invitations handled within this campaign during the interval (whether sent from this campaign or not).	int	Not Null
chatinvitationsexpired	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

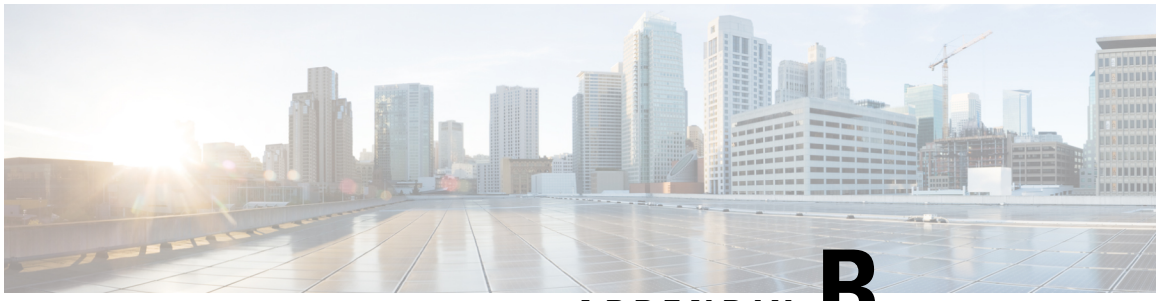
The mmca_agent_campaign_activity table is an aggregate table used for reporting agent-related campaign statistics.

Table 3: mmca_agent_campaign_activity

Field name:	Description:	Data type:	Constraints:
recordid	Auto-incrementing ID	serial(8)	Primary key, not Null.
interval	Reporting interval to which this record applies (currently only a 15 minute interval is supported).	datetime	Not Null.
campaignid	ID of the campaign from the mmca_reportcampaign table.	int(8)	Not Null.
userid	String representing the login name of the user who modified this record for this interval.	varchar	Not Null.
schandled	Number of social contacts handled in this interval by the userid for this campaign during this interval.	int	Not Null.
scdiscarded	Number of social contacts discarded in this interval by the userid for this campaign.	int	Not Null.

Field name:	Description:	Data type:	Constraints:
screserveddiscarded	Number of discarded social contacts in this interval that were previously reserved (at any time) by this user.	int	Not Null.
screservedhandled	Number of handled social contacts in this interval that were previously reserved (at any time) by this user.	int	Not Null.
handledtime	Cumulative handled time for all social contacts handled by this user in this interval for this campaign. 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.
discardedtime	Cumulative discarded time for all social contacts discarded by this user in this interval for this campaign. Discard time is defined as the time between when a contact was marked as reserved and the time the contact was marked discarded.	int(8)	Not Null.
chatinvitationssent	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.

Field name:	Description:	Data type:	Constraints:
chatinvitationshandled	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.
chatinvitationsexpired	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.



APPENDIX B

SocialMiner Server Configuration

• Security Configuration Options , on page 205

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.

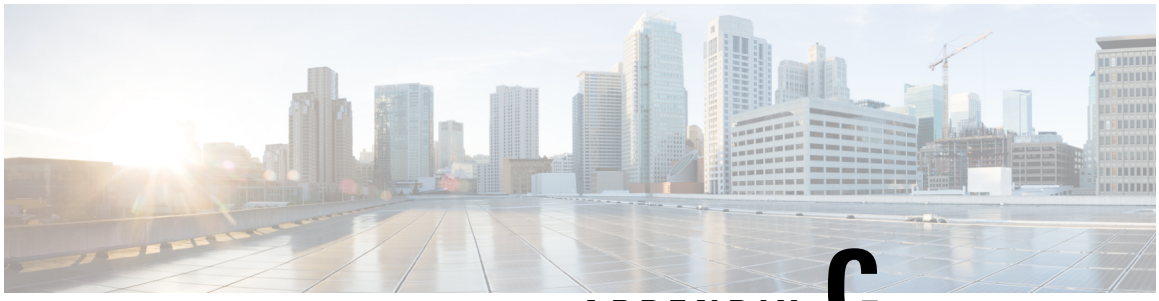
SocialMiner recognizes the following reverse proxy headers:

Header	Required?	Comments
X-Forwarded-Host	Y	Includes the proxy host name as visible to the user. May also include a port in the form <server name>:<port>

Header	Required?	Comments
X-Forwarded-Proto	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.
X-Forwarded-Port	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.
Front-End-https	N	If present and value is "on", returned links will use https. It overrides X-Forwarded-https when present.
X-Forwarded-https	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"
```



APPENDIX C

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, on page 207

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.

ccp.campaign.updates

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.

Note This node does not apply to Task Routing tasks submitted through the Task API. For more information on status polling with the Task API, see the XMPP Eventing Versus Task API Polling, on page 70

ccp.serviceability.eventingInfo

The ccp.serviceability.eventingInfo node allows applications to receive a notification when the SocialMiner web server establishes a connection to the SocialMiner XMPP service. In failure scenarios, this event provides applications with an XMPP channel to notify those applications that the SocialMiner web server is running.

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.

Field	Description
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>https://[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.
extensionFields	A collection of custom name and value pairs.

XMPP BOSH Eventing

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 social contact's statusReason is modified.
- An email contact is requeued. When an email contact is requeued, the statusReason is EMAIL_REQUEUE_TRANSFER or EMAIL_REQUEUE_AGENT_DISCONNECTED.

The actual payload of the XML event is as follows:

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

Serviceability Events

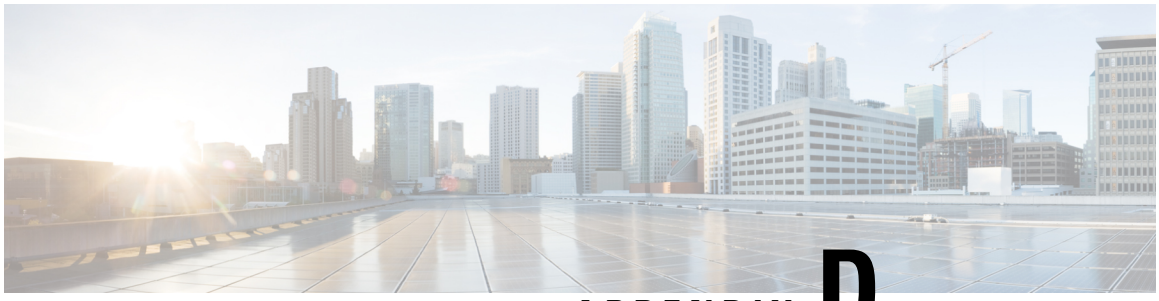
Events on the ccp.serviceability.eventingInfo node contain the following attributes:

Field	Description
connectionStatus	The status of the eventing subsystem XMPP connection. A status of CONNECTED means that the eventing subsystem is connected to the XMPP server.

The actual payload of the XML event is as follows:

```
<Serviceability xmlns="https://jabber.org/protocol/pubsub">
  <connectionStatus>CONNECTED</connectionStatus>
</Serviceability>
```

XMPP BOSH Eventing



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, on page 213

Javascript Concepts

Gadgets API

Custom reply templates are instances of OpenSocial gadgets (see https://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 https://<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 https://<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.