Using the Audit Log

Introduction

The Audit Log is a tool that allows site administrators to view actions performed by users and administrators via the Console, or by end users to their own accounts (e.g., changing a password). All actions performed from within the console are audited, as well as end-user actions involving the APIs listed below. API calls using application keys *are not* audited, except where otherwise noted.

Audited events are stored for one year from the date they occurred.

Although all audited events are logged, they may not appear in the Audit Log if the user/group viewing the page doesn't have the necessary privileges. These privileges may restrict viewing items at the site level, or allow viewing items on a global, partner level.

Watch an Instructional Video

If you have an SAP logon, you can watch an instructional video about the Audit Log here.

> The Audit Log is a feature of the 'Identity Enterprise' package, which is a premium service that requires activation. If it is not part of your site package, please contact Gigya Support via the Console .

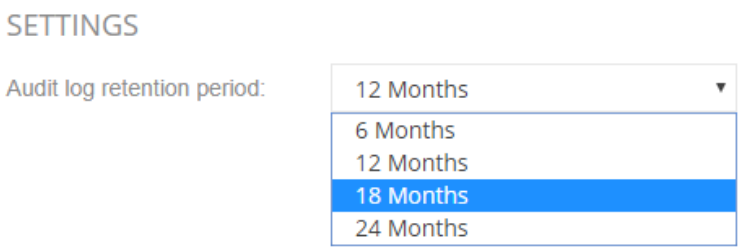
Configuration

You can configure the retention period for saving audit log records. This configuration affects both the Audit Log, and the Account Audit Log.

By default, this period is set to 12 months, so if that suits your needs, no additional configuration is required.

To change this setting:

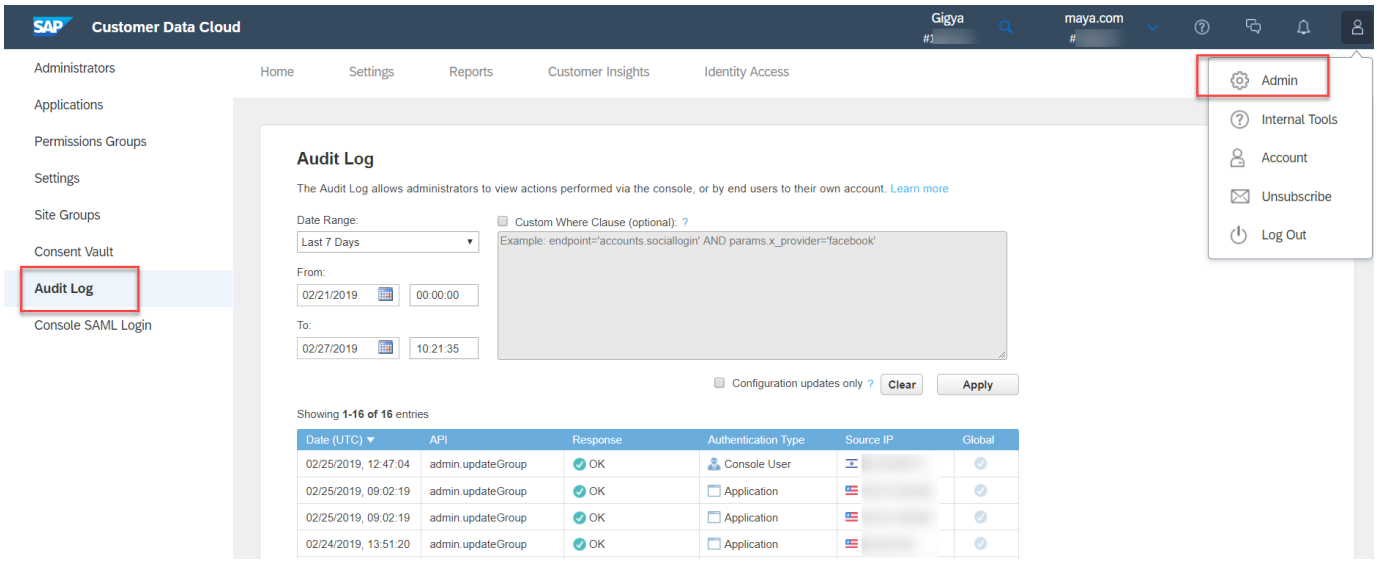
1. Go to the Admin tab in Gigya's Console and select Settings.
2. Under **Audit log retention period**, select the number of months that audit records will be stored.



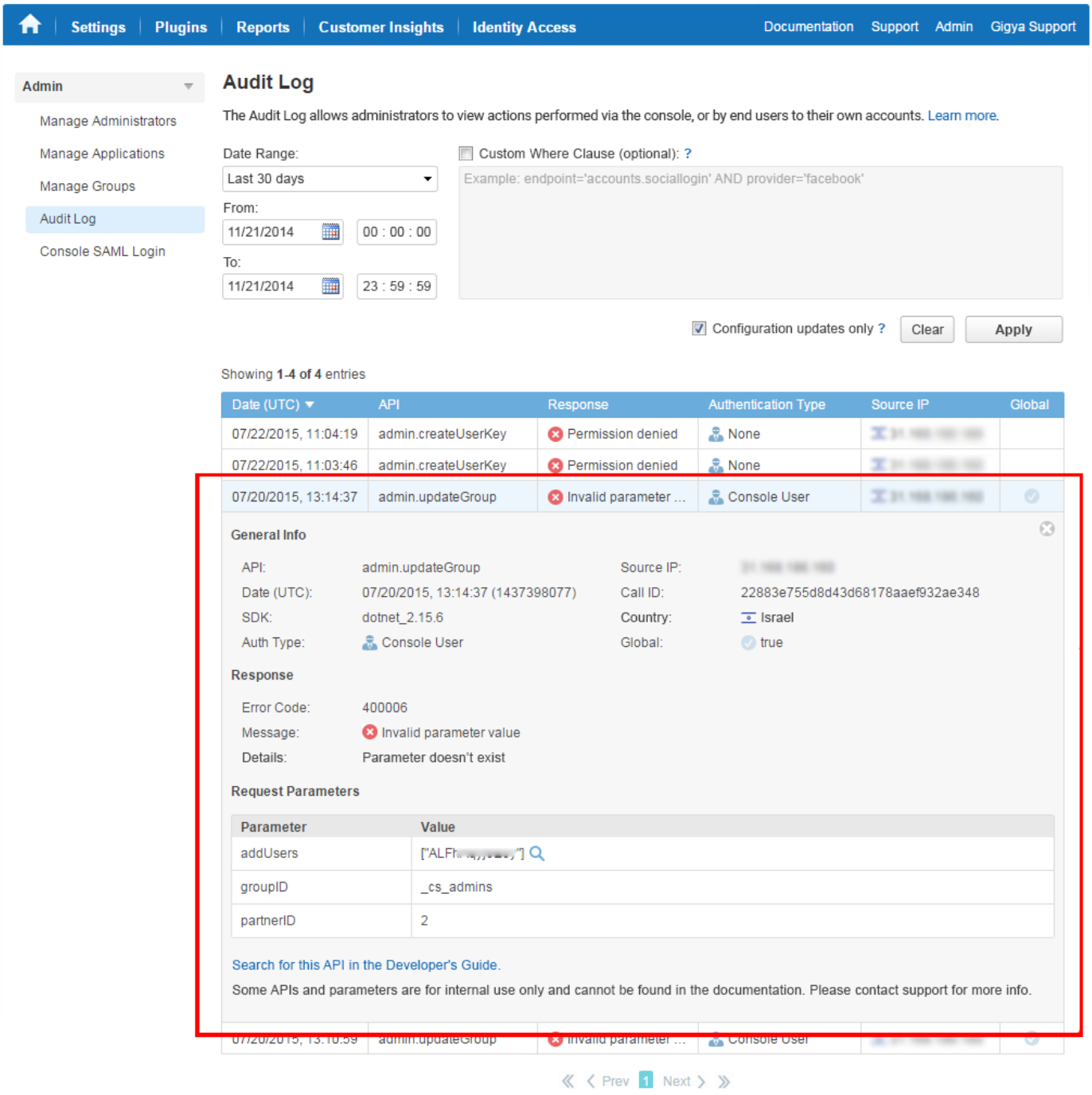
3. Save and confirm.

The Audit Log

Open the **Admin** menu from your user menu, and select **Audit Log** on the left hand side:

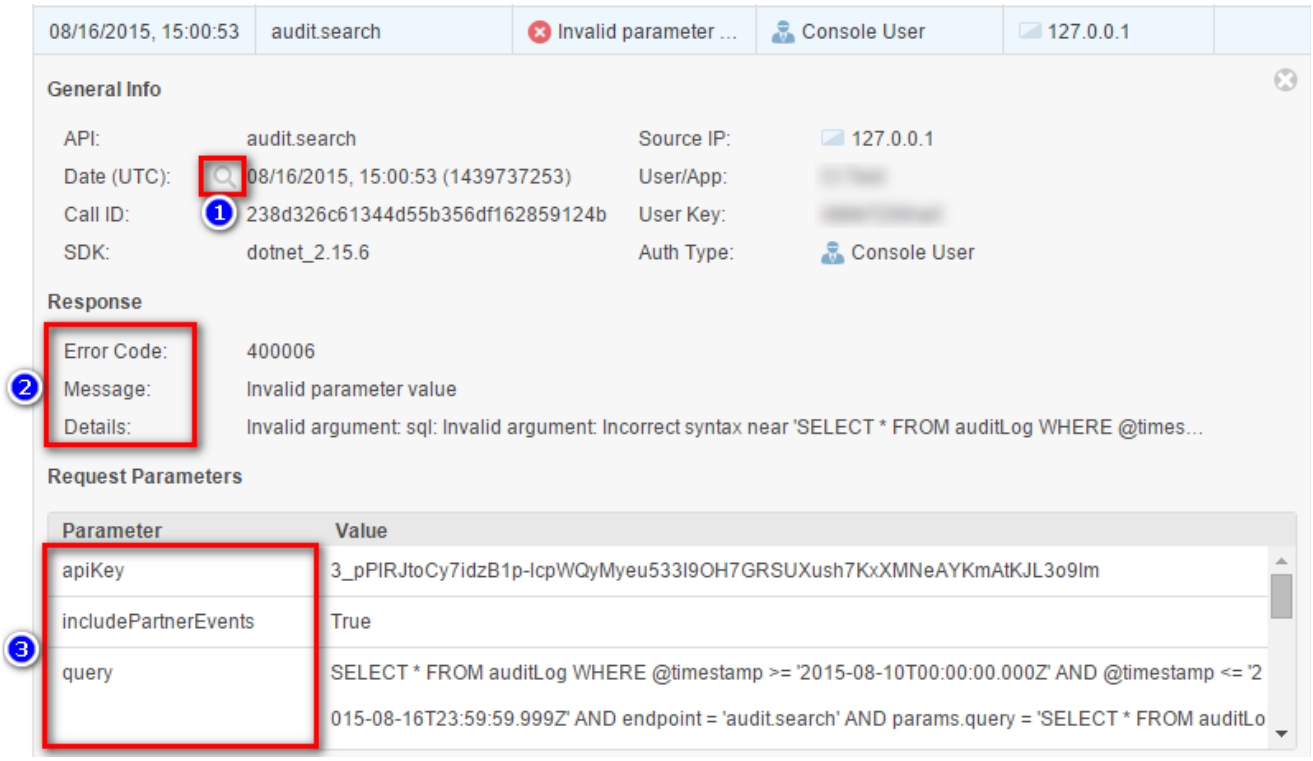


Clicking a log entry expands it to display extended information:



The Extended Information Panel

The extended information panel contains a number of features to help you make the most of your log data. In addition to basic data about the API call, the panel includes:



1. **The Magnifier:** Appears while hovering over any field value which allows filtering via the Advanced Query tool. Clicking the magnifier will automatically add the current field and value to the Advanced Query box and submits the current query.
2. **The Response:** Appears for every API call and contains the error code, error message and error details (in case of an error). If no error occurred, the Details field does not appear.
3. **The Request Parameters:** Appears for every API call and contains the list of parameters (including values) submitted with the request.

Note: When the User/App field contains 'Gigya Admin' it means that the current log entry refers to an action performed by Gigya.

Advanced Queries

The Audit Log includes an Advanced Query tool which allows you to view audit log entries using SQL syntax. See the audit.search documentation for a complete explanation on supported SQL operations and syntax. It's important to note that the advanced query is a WHERE clause that is automatically appended to the selected dates and the *Configuration updates only* option. That is, the *SELECT ..... FROM .... WHERE* portion of the query is automatically implied.

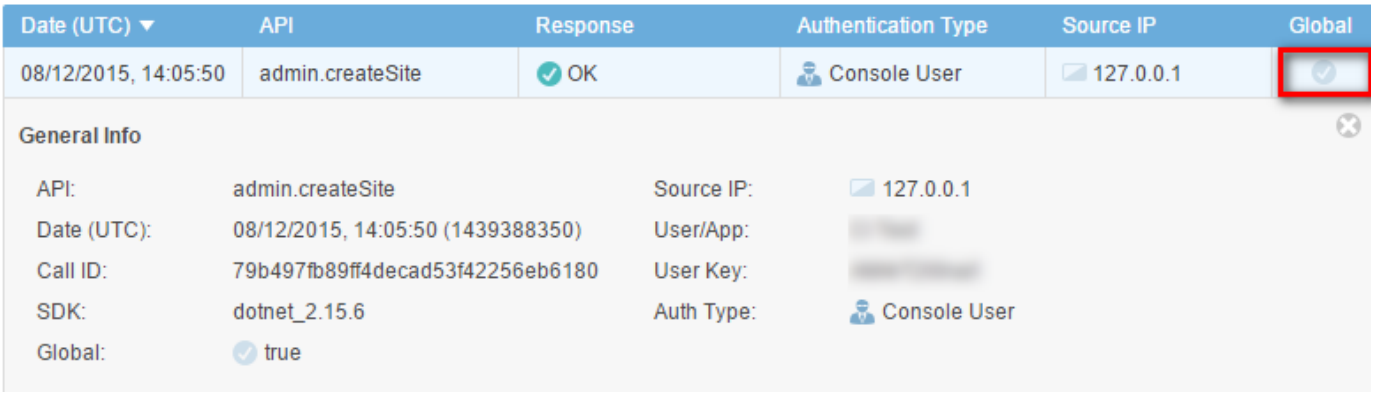
For example, a query for all actions performed via any *socialize* API between Aug 16, 2015 and Aug 17, 2015:

```
endpoint LIKE 'socialize%' AND @timestamp > '2015-08-16T00:00:00.000Z' AND
@timestamp < '2015-08-17T00:00:00.000Z'
```

> Encrypted fields, such as userKeyDetails.name, are not searchable, and trying to query a specific value for these fields will not return results.

Global Entries

When an entry is marked Global, it designates an API call made outside the scope of a specific site. These APIs are used to create sites, get user and group information, get and set ACLs, and more.



Configuration Updates Only

Selecting the *Configurations updates only* option filters the Audit Log by 'get', 'set' and 'update' APIs, even when using advanced queries. The following APIs are displayed when *Configuration updates only* is selected:

Accounts	accounts.deleteScreenSet		***GM***	gm.setActionConfig
	accounts.registerCounters			gm.setChallengeConfig
	accounts.setSchema			gm.setGlobalConfig
----------	----------		----------	----------
IDS	ids.registerCounters		***Comments***	comments.setCategoryInfo
	ids.setSchema			comments.setStreamInfo
	ids.unregisterCounters		----------	----------
----------	----------			
DS	ds.setSchema			
FIDM.SAML	***Gigya as SAML SP***			
	fidm.saml.delIdP		The license could not be verified: License Certificate has expired!	The license could not be verified: License Certificate has expired!
	fidm.saml.importIdPMetadata		The license could not be verified: License Certificate has expired!	The license could not be verified: License Certificate has expired!
	fidm.saml.registerIdP			

Audited APIs

The following APIs are audited in addition to all actions performed via the Console. This means that any action performed by end users via these APIs will appear in the Audit Log.

- Any actions performed using an application key are NOT audited, unless otherwise noted.
- Any actions performed using a user key and secret ARE audited. This means an API that is not on the list, may appear in the audit log.

Accounts			***Socialize***	socialize.addConnection
	accounts.deleteAccount *application keys audited			socialize.login
	accounts.deleteSchemafields			socialize.deleteAccount *application keys audited
	accounts.deleteScreenSet			socialize.logout
	accounts.finalizeRegistration			socialize.notifyLogin
	accounts.importLiteAccount			socialize.removeConnection
	accounts.initRegistration *audited only on server-side calls, and not when using it client-side or in the web SDK			socialize.setProviderConfig

	accounts.isAvailableLoginID			socialize.setUID
	accounts.linkAccounts			
	accounts.login (appears as socialize.login) *application keys audited		---	---
	accounts.logout		***FIDM.SAML***	***Gigya as SAML SP***
	accounts.notifyLogin			fidm.saml.delIdP
	accounts.rba.setPolicy			fidm.saml.getConfig
	accounts.rba.unlock			fidm.saml.getRegisteredIdPs
	accounts.register *application keys audited			fidm.saml.importIdPMetadata
	accounts.removeConnection			fidm.saml.registerIdP
	accounts.resetPassword			fidm.saml.setConfig
	collapsed			
	accounts.setAccountInfo *application keys audited		---	---
	accounts.setPassword		***FIDM.SAML***	***Gigya as SAML IdP***
	accounts.setPolicies			fidm.saml.idp.delSP
	accounts.setUID			fidm.saml.idp.getConfig
	accounts.setSchema			fidm.saml.idp.getRegisteredSPs
	accounts.setScreenSet			fidm.saml.idp.importSPMetadata
	accounts.socialLogin (appears as socialize.login)			fidm.saml.idp.registerSP
	accounts.tfa.*			fidm.saml.idp.setConfig
	accounts.verifyEmail			
	The license could not be verified: License Certificate has expired!			
---	---		---	---
Admin	All APIs in the admin namespace are audited. Click for a list of Admin APIs **admin.certificates.cancelCurrentRequest** • This API cancels any SSL certificate currently in **Pending** status. **admin.certificates.createRequest** • This API initiates the flow for a new Domain Proxy SSL certificate. **admin.certificates.finalizePendingRequests** • This API finalizes all pending certificate requests and publishes them to CloudFront. **admin.certificates.getCurrentRequestStatus** • This API returns the current status of a given site's certificate. **admin.certificates.resendVerificationEmails** • This API resends verification emails for the requested certificate domains to the contact addresses on file with whois. **admin.clearCache** • This method clears the cache for all sites according to the keyPrefix. **admin.console.finalizeTenantInvitation** • This API finalizes the Invitation of a new console partner. **admin.console.getApiKey** • This API returns the API key for the Primary data center the partner is being created on based upon the data center receiving the request. **admin.console.isTokenValid** • This API returns if the submitted token is valid (Boolean). **admin.createGroup** • This method creates a new group. **admin.createPartner** • This API creates a new partner record, and sets the secret key and encryption key for it. **admin.createSite** • This method creates a new site. **admin.createUserKey** • This API creates a new user key and user secret pair. **admin.deleteACL** • This API deletes a previously-saved ACL. **admin.deleteGroup** • This API deletes an existing group. **admin.deleteSite** • This API deletes an existing site by API key. **admin.deleteUserKey** • This API deletes an existing user and removes them from all groups to which they belong.		***DS***	ds.deleteSchemaFields

admin.getACL

- This API retrieves a partner's previously-saved ACL and its description, or a built-in ACL.

admin.getEffectiveACL

- This API returns the effective permissions of a certain user for a specific partner, and optionally site, or returns the effective permissions of an arbitrary list of a specific partner's groups.

admin.getGroups

- This API returns a single group if specified, or all of a partner's groups.

admin.getGroupUsers

- This API retrieves the users of an existing group.

admin.getPartner

- This API retrieves a specified partner's information.

admin.getPartnerSites

- This API retrieves all existing sites of a partner, including all the site's configured settings.

admin.getRestrictions

- This API retrieves the comments restrictions of a specified site.

admin.getSiteConfig

- This API retrieves the configuration of existing sites.

admin.getUserSites

- This API returns either:
    1. All sites with which a user is associated by way of group memberships. This is computed by looking at the scopes of all the groups to which a user belongs. We do not include groups whose scope applies to all partners. We look only for group membership, we do not check whether the user has certain permissions in these groups.
    2. All sites in a specific partner with which a user is associated, even through a group that applies to all partners.

admin.resetSecretKey

- This API revokes the existing, and creates a new, secret key for the specified partner.

admin.search

- This method searches the partner IDs, site IDs, base domains, and company names.

admin.setACL

- This method creates or updates an existing ACL.

admin.setRestrictions

- This method sets the comments restrictions for the site.

admin.setSiteConfig

- This method sets the configuration for existing sites.

admin.tenant.create

- This API initiates a new Tenant object for the supplied tenantID.

admin.tenant.delete

- This API deletes the requested tenant.

admin.tenant.getAll

- This API returns all existing tenants for the current data center.

admin.tenant.getAllSites

- This API returns all partner and site IDs linked to the tenantID submitted.

admin.tenant.getInvitationLink

- This API returns the invitation link for a new partner.

admin.tenant.get

- This API returns the existing metadata for the submitted tenantID.

admin.tenant.update

- This API updates an existing tenant's metadata.

admin.updateGroup

- This method updates an existing group.

admin.updatePartner

- This method updates a partner's information, including enabling and disabling features and services.

admin.updateUserKey

- This method allows a user to update the name and email address associated with a userKey.

Please note that events may or may not appear in the audit log, depending on the privileges granted to the user/group viewing the log.

No APIs are audited when the errorCode is one of the following:

- 400002 - Missing_required_parameter
- 400006 - Invalid_parameter_value
- 400093 - Invalid_ApiKey_parameter
- 400096 - Not_supported
- 403005 - Unauthorized_user
- 403007 - Permission_denied
- 403048 - Api_rate_limit_exceeded
- 403210 - Deleted_API_Key

Additional Information

Operations performed by a Lite account (opposed to a Full Registered User) can be distinguished by:

- `"authType" : "liteRegToken"`

Any events occuring via an OIDC RP are logged like any other social network interaction. OIDC OP events are not audited.

The license could not be verified: License Certificate has expired!