Webhooks

A webhook is a user-defined HTTPS callback that is triggered when a particular event occurs in a KB record. The webhook receives real-time notification messages for events. When the event happens, the service makes an HTTPS POST request to the webhook's HTTPS URL.

Webhooks are useful when you want to receive and process real-time data updates. Instead of needing to make a REST API call to check for changes, when you register webhooks in your KB, the service automatically pushes HTTPS POST event notifications to the webhook's URL to tell the webhook the event has occurred. This push-based model requires fewer API requests overall and provides real-time updates with simpler build. With webhooks, you can build more robust apps and update your application instantly.

Agiloft webhook notifications are generated when a record is created, edited, or deleted. The event notification includes updated information about the field values that were selected in your webhook configuration. Optionally, these notifications can also include detailed information like status changes.

To create and initialize a webhook:

1. Either the client application calls POST `/ewws/webhooks` with a user token to register the webhook as a resource, or you create a webhook manually in **Setup > Integration > Webhooks Setup**.
2. Agiloft validates that the POST request is valid and the webhook URL is valid. There is a special processing for this validation where Agiloft makes the HTTPS GET request to your webhook URL, and the webhook URL is expected to respond in a specific way. For details, see Verifying the Webhook URL.
3. Agiloft sends the success response with an HTTP 2XX code to your client application (if you are using a REST API call) with a unique webhook identifier and Location header, which contains the URL of the webhook resource. This URL is created in Webhooks Setup in your KB.
4. Going forward, whenever the specified event happens in your KB, a notification for that event is sent to the webhook URL.

Prerequisites

If you want to register a webhook using an API call, instead of creating the webhook manually in **Setup > Integration > Webhooks Setup**, the call requires an OAuth access token, JWT token, or user credentials. Each webhook event (Create/Update/Delete) requires the same permissions that grant access to Webhooks Setup in your KB, and your application must have all these required permissions during this process. Typically, the admin group has these permissions, but you might grant other groups permission to access Webhooks Setup.

If the user doesn't have the appropriate permissions and a webhook REST request is sent on his behalf, that request must register or modify the user's own webhook entries. To do so, it must specify the correct user login for the "User’s Permissions to Use" parameter (see table below). The value of this parameter must match the username from which the webhook REST request came.

Whichever authentication method you use, you must also use it in the following REST endpoints to perform operations on behalf of the user who authorized the API access.

Configuring Webhooks

Webhooks can be created using REST APIs, or built directly in your KB from **Setup > Integration > Webhooks Setup**. When you access it for the first time, Webhooks Setup shows a Deploy button; after you click that and webhooks are successfully deployed, click Configure to access webhooks. The table is empty to begin with, but going forward, any webhook you create will appear in this list, whether it was created manually in Webhooks Setup or registered using a REST API call.

Underscores

Do not use underscores in webhook headers.

Each webhook entry sets the following properties:

Parameter	Value	Required	Description
ID	Automatically generated	yes	Identifies the webhooks entry in the webhooks configuration table.
Webhook Key	Automatically generated	yes	Unique identifier of webhook. An identifier is generated automatically when a webhook is successfully registered, using either a REST API call or by saving a new entry in the Webhooks Setup menu. This identifier must be used in REST webhook API calls.
Status	Active /Inactive	yes	Status of webhook. A webhook can be either active or inactive. By default, a webhook is inactive. An Active webhook will receive requests for events as they occur within KB. A webhook marked as Inactive will stop receiving event requests. Any other existing, unprocessed event requests will be canceled and not sent to your webhook endpoint. If an inactive webhook is made active, it will begin receiving event requests once more as soon as new event notifications occur.
Title	Any string	no	Title of your webhook.
Description	Any string	no	Description of the webhook.

Webhook URL	Https URL	yes	Webhook's HTTPS URL. When the webhook event occurs, the service makes an HTTPS POST request to this URL. The webhook is an HTTPS-based service that listens at a specific URL for incoming HTTPS POST notifications that are triggered when events occur in the table entries within your KB. You need this URL to subscribe your webhook to the event notifications. A webhook HTTPS URL must be accessible all times so that Agiloft can send a POST request. This URL must be available on the public Internet. Make sure your webhook supports POST requests for incoming notifications and GET requests for the verification process (see Verifying the Webhook URL section). Your webhook URL must not be blocked by a firewall.
Table Name	Table name from the KB	yes	Name of the table where you want to track changes.
Fields to Retrieve	Any field names	no	The field names you want the webhook to send. Fields should be listed using the name, not the label, and with each field delimited by a comma: contract_title0, company_name, created_by. Supported types of fields: Append Only Text, Calculated Result, Choice, Currency, Date, Date/Time, Email, Floating Point, Integer, Long Integer, Multi Choice, Percentage, Short Text, Telephone/Fax, Text, Time, URL, Linked field (only fields which can be converted to a text value). Other field types can't be sent as a part of webhook body and they must be retrieved directly by REST API.
Event type	Create /Update /Delete	yes	The type of event you want to track in table entries. Select Create to track new table entries, created from scratch; select Delete to track when users delete entries from the table; or select Update to track all updates to entries in the table.
Modification type	Email/Web /API	no	Choose what types of record updates will trigger the webhook. By default, the value is empty, which will trigger the webhook without tracking the type of entry modification. The following types of entry modification are possible: ▪ Email: all updates that are made by inbound email ▪ Web: user interactions with records through the web portal ▪ API: record edits by the REST API
User's Permissions to Use	The login of an existing user	yes	The webhook call uses the permissions of the user you select here. If you select a user who has limited permissions, that can affect what is passed in the call. For example, if this user doesn't have permission to view a certain record, or fields within the record, that content will not be passed in the webhook call.

State	Any string	recommended	This value will be returned to the client as a header. While not required, use of the state key parameter is highly recommended to protect against CSRF. Maximum length of key is 255 symbols.
Webhook Confirmation	Yes/No	no	Use confirmation in webhooks. If set to Yes, or not specified, webhooks will work in standard mode, where the Verification-Code should be included in the answer when you receive an incoming webhook event. If set to No, then when you receive a webhook event, you only need a response with a 2XX status code on an incoming webhook event, rather than needing to provide confirmation by Verification-Code. For more information about Verification-Code and verification, see Verifying the Webhook URL below.
Processing by the Action	Yes/No	yes	Default value is set to No. If set to No, the webhook is processed using the standard workflow described above. This means that process of real-time data updates happening when table entry is created/edited/deleted will be processed at the same time when event is occurs. If set to Yes, the webhook event will not be processed by the standard workflow. Instead, the event notification will be sent by the Rule only. Real-time processing of this webhook will be ignored. This means that the notification will be sent out when the condition in a Rule is matched and when a Rule triggers the webhook action directly.
Record Filter	Any string	no	Use a record filter in webhooks. This accepts the same format used in the Search REST call, although saved searches cannot be used here. An example of a query: company_name=Agiloft || use_as_reference=No The following operators are supported: ▪ Equals: == ▪ Does not equal: != ▪ And: && ▪ Or: || ▪ Less than: < ▪ Less than or equal to: <= ▪ Greater than: > ▪ Greater than or equal to: >=
Date Created	Automatically generated	yes	The system timestamp showing when the webhook entry is created.
Owner	Automatically generated	yes	Owner of the webhook entry.

Creating Webhooks in Agiloft

You can create a new webhook directly in Agiloft. To do so:

1. Go to **Setup > Integration > Webhooks Setup** and click Deploy, then click Configure when the deployment is complete.
2. In the Webhooks Setup page that opens, click New to create a new webhook.
3. Complete all the fields, which correspond to each of the parameters described above.
4. Once you have entered all required data, click Save.
5. To activate your webhook, click Enable.

When you enable a new webhook, Agiloft makes the HTTPS GET request to your webhook URL. The webhook URL is expected to respond in a specific way. If the URL responds correctly, Agiloft automatically makes the webhook Active, and it's ready for you to use.

If you need to deactivate the webhook later, simply click Disable for that entry in the list. Always disable a webhook before you delete it in Webhooks Setup. Otherwise, if you remove the entry from the list without deactivating the webhook, the webhook remains active until the next time the KB restarts.

✓ If you need to change the settings for a webhook that is already active, make sure to click Disable before you make those changes. After you finish updating the settings, save the changes and click Enable to re-activate the webhook.

Creating Webhooks with REST API

You can also create webhooks using a REST API call instead of the Webhooks Setup menu.

This can be done by calling the POST `/ewws/webhooks` API from your application and passing along the subscription data. The subscription specifies how the webhook intends to consume events. For details about constructing the API call, see the parameters described in the Register API section.

When Agiloft receives a webhook creation request from your application, the Webhook service first verifies the webhook URL by making a GET call it. If that call succeeds, the Webhook service returns the response of the POST call, with the URL of the newly created webhook resource contained in the HTTPS Location header. The client service can later make PUT/GET/DELETE calls on this URL. See the list of API calls for webhook management below.

Verifying the Webhook URL

Before registering a webhook configuration successfully, Agiloft verifies that the provided webhook URL is able to receive event notifications. To test this, when a new webhook registration request is received, Agiloft makes a verification request to the webhook URL. This verification request is a HTTPS GET request. This request has a custom HTTP header, Verification-Code. The value in this header is set to the random generated code that is requesting to create and register the webhook. To successfully register a webhook, the webhook URL must respond to this verification request with the 2XX response code in one of two ways:

- In a response header, Verification-Code. This is the same header which was passed in the request, and can be echoed back in the response.
- In JSON response body with the key of Verification-Code, its value being the same as in the Verification-Code header that was sent in the request.

The webhook is registered only if the URL successfully sends the 2XX response code as expected. If you accidentally enter an incorrect webhook URL, the URL will fail to respond correctly to the verification of intent request, and Agiloft will not send any notifications to that URL.

In addition, the webhook URL can also validate that it would receive notifications only through the webhooks which are registered by a specific application. This can be done by validating the verification codes that are passed in the Verification-Code header. If the webhook URL does not recognize that verification code, it must not respond with the success response code, and Agiloft will take care that the provided URL is not registered as a webhook.

Webhook URL calls are always verified when:

- Creating (registering) the Webhook entry. If this verification of webhook URL call fails, the webhook will not be created.
- Updating (changing) the Webhook entry. This changes a webhook's status from Inactive to Active. If this verification of webhook URL call fails, the webhook status will not be changed to Active.

Responding to Webhook Notifications

The Agiloft webhook service performs an implicit verification of the webhook URL in each webhook notification request that is sent, as long as the Webhook Confirmation parameter is set to Yes or left blank. Therefore, every webhook notification HTTPS request also contains the custom HTTPS header called Verification-Code. The value of this header is the same value of Verification-Code (the random generated key) as described in Verifying the Webhook URL.

The Webhook service considers the webhook notification successfully delivered using the same criteria as the initial registration. Otherwise, the Webhook service reattempts delivery of the notification to the webhook URL up to five times, at 10 seconds, 30 seconds, five minutes, 15 minutes, and 40 minutes.

Format of Error Responses

Each response has an HTTPS error status, error code, and error description. The error responses from the webhook service have the following Json format: `{ "error" : "", "error_description" : "" }`

Standard Error Codes

This list covers common error codes, but not all. Note that new error codes might be added later, and existing error codes might be updated. Your application should be prepared to do default handling for error scenarios.

Status code	Error type	Error description
400	BAD_REQUEST	The request provided is invalid.
400	INVALID_JSON	An invalid JSON was specified.
400	MISC_ERROR	Some miscellaneous error has occurred.
401	UNAUTHORIZED	Client must authenticate itself to get the requested response.
401	INVALID_ACCESS_TOKEN	An access token provided in the request is invalid or has expired.
401	INVALID_LOGIN	The login provided in the request is invalid or has blocked.
403	FORBIDDEN	The client does not have access rights to the content; that is, it is unauthorized, so the server is refusing to give the requested resource.
405	METHOD_NOT_ALLOWED	The request method is known by the server but is not supported by the target resource.
500	SERVER_ERROR	Some miscellaneous server error has occurred.

Webhook REST API

This section details the REST API calls specific to using webhooks.

Registering Webhooks

The client application makes an HTTPS POST call to the `/ewws/webhooks` endpoint, using an OAuth access token, JWT token, or user credentials as authentication, and including the following parameters:

Object	Required	Value	Notes
Request Type	-	POST	-
Request Content-Type	yes	application/json	-
Request body	yes	{ "title": "", "description": "", "webhook_url": "", "webhook_status": "", "table_name": "", "event_type": "", "modification_type": [""], "entry_fields": [""], "record_filter": "", "user_permissions": "", "webhook_confirmation": "", "state_key": "", "by_rule": "" }	Title, description, URL, status, table name, event type, and user permissions are required. For more information about the parameters, see Configuring Webhooks
Response Content-Type	-	application/json	-
Response header	-	string	Location Header specifying the resource location of the webhook

Response body	-	{
		"id": integer value,
		"webhook_key": ""
		}
			For more information about the parameters, see Configuring Webhooks

Response status code	-	201	The server must respond with this status.

Error Codes

This list covers common error codes, but not all. Note that new error codes might be added later, and existing error codes might be updated. Your application should be prepared to do default handling for error scenarios.

Status code	Error type	Error description
400	INVALID_PARAMETERS	Some parameters in the request are invalid.
400	INVALID_URL	An invalid webhook URL was specified.
400	MISSING_REQUIRED_PARAM	The required parameters are missing.
400	WEBHOOK_LIMIT_EXCEEDED	This webhook can't be created. The events array {events} has reached the maximum number of active webhooks.
403	WEBHOOK_CREATION_NOT_ALLOWED	Webhook creation is not allowed.

See also the Format of error response section for details.

Retrieving Webhook Information by Webhook Key

The client application makes an HTTPS GET call to the `/ewws/webhooks` endpoint, using an OAuth access token, JWT token, or user credentials as authentication, and including the following parameters:

Object	Required	Value	Notes
Request Type	-	GET	-
Request Parameter called wh_key	yes	The Webhook Key of your webhook entry	URL-encoded value of your Webhook Key you want to retrieve

Response body	-	{ "webhook_id": integer value, "webhook_key": "", "title": "", "description": "", "webhook_url": "", "webhook_status": "", "table_name": "", "event_type": "", "modification_type": [""], "entry_fields": [""], "record_filter": "", "user_permissions": "", "webhook_confirmation": "", "state_key": "", "by_rule": "", "owner": { "login": "", "full_name": "" } }	For more information about the parameters, see Configuring Webhooks
Response Content-Type	-	application/json	-
Response status code	-	200	The server must respond with this status.

Error codes

This list covers common error codes, but not all. Note that new error codes might be added later, and existing error codes might be updated. Your application should be prepared to do default handling for error scenarios.

Status code	Error type	Error description
400	INVALID_PARAMETERS	Some parameters in the request are invalid.
400	MISSING_REQUIRED_PARAM	The required parameters are missing.
404	INVALID_WEBHOOK_KEY	An invalid webhook key was specified.

See also the Format of error response section for details.

Retrieving the List of Webhooks

The client application makes an HTTPS GET call to the `/ewws/webhooks` endpoint, using an OAuth access token, JWT token, or user credentials as authentication, and including the following parameters:

Object	Required	Value	Notes
Request Type	-	GET	-
Query Parameters	yes	▪ byStatus: Active, Inactive, All. Query parameter to fetch all active webhooks (if Active value is specified) or all inactive webhooks (if Inactive value is specified) or all entries in other case. The query parameter is optional. ▪ byEventType: Create, Update, Delete. Query parameter to fetch all webhooks by event type. The query parameter is optional. ▪ byTable: the_table_name. Query parameter to fetch all webhooks by table name within KB. The query parameter is optional. ▪ byField: the_field_name. Query parameter to fetch all webhooks by field name within KB. The query parameter is optional.	Query parameter values should be URL-encoded.

Response body	-	[{
		"webhook_id": integer value,
		"webhook_key": "",
		"title": "",
		"description": "",
		"webhook_url": "",
		"webhook_status": "",
		"table_name": "",
		"event_type": "",
		"modification_type": [
		""
		],
		"entry_fields": [
		""
		],
		"record_filter": "",
		"user_permissions": "",
		"webhook_confirmation": "",
		"state_key": "",
		"by_rule": "",
		"owner": {
		"login": "",
		"full_name": ""
		}
		},...]
			For more information about the parameters, see Configuring Webhooks

Response Content-Type	-	application/json	-
Response status code	-	200	The server must respond with this status.

Error codes

This list covers common error codes, but not all. Note that new error codes might be added later, and existing error codes might be updated. Your application should be prepared to do default handling for error scenarios.

Status code	Error type	Error description
400	INVALID_PARAMETERS	Some parameters in the request are invalid.
400	MISSING_REQUIRED_PARAM	The required parameters are missing.

See also the Format of error response section for details.

Updating the Webhook

The client application makes an HTTPS PUT call to the `/ewws/webhooks` endpoint, using an OAuth access token, JWT token, or user credentials as authentication, and including the following parameters:

Object	Required	Value	Notes
Request Type	-	PUT	-
Request Content-Type	yes	application/json	-
Request body	yes	{ "webhook_key": "", "title": "", "description": "", "webhook_url": "", "webhook_status": "", "table_name": "", "event_type": "", "modification_type": [""], "entry_fields": [""], "record_filter": "", "user_permissions": "", "webhook_confirmation": "", "state_key": "", "by_rule": "" }	The webhook key is required, and the other parameters are optional. For more information about the parameters, see Configuring Webhooks
Response Content-Type	-	application/json	-
Response body	-	{ "id": integer value, "webhook_key": "" }	For more information about the parameters, see Configuring Webhooks
Response status code	-	200	The server must respond with this status.

Error codes

This list covers common error codes, but not all. Note that new error codes might be added later, and existing error codes might be updated. Your application should be prepared to do default handling for error scenarios.

Status code	Error type	Error description
400	INVALID_PARAMETERS	Some parameters in the request are invalid.
400	INVALID_URL	An invalid webhook URL was specified.
400	MISSING_REQUIRED_PARAM	The required parameters are missing.
403	WEBHOOK_CREATION_NOT_ALLOWED	Webhook creation is not allowed.

See also the Format of error response section for details.

Activate or Deactivate the Webhook

The client application makes an HTTPS PUT call to the `/ewws/webhooks` endpoint, using an OAuth access token, JWT token, or user credentials as authentication, and including the following parameters:

Object	Required	Value	Notes
Request Type	-	PUT	-
Request Content-Type	yes	application/json	-
Request body	yes	{ "webhook_key": "", "webhook_status": "" }	Both parameters are required. For more information about the parameters, see Configuring Webhooks
Response Content-Type	-	application/json	-
Response body	-	{ "id": integer value, "webhook_key": "" }	For more information about the parameters, see Configuring Webhooks
Response status code	-	200	The server must respond with this status.

Error codes

This list covers common error codes, but not all. Note that new error codes might be added later, and existing error codes might be updated. Your application should be prepared to do default handling for error scenarios.

Status code	Error type	Error description

400	INVALID_PARAMETERS	Some parameters in the request are invalid.
400	MISSING_REQUIRED_PARAM	The required parameters are missing.
403	WEBHOOK_CREATION_NOT_ALLOWED	Webhook creation is not allowed.

See also the Format of error response section for details.

Deactivating and Removing the Webhook

The client application makes an HTTPS DELETE call to the `/ewws/webhooks` endpoint, using an OAuth access token, JWT token, or user credentials as authentication, and including the following parameters:

Object	Required	Value	Notes
Request Type	-	DELETE	-
Request Content-Type	yes	application/json	-
Request body	yes	{ "webhook_key": "" }	For more information about the parameters, see Configuring Webhooks
Response body	-	Empty	-
Response status code	-	200	The server must respond with this status.

Error codes

This list covers common error codes, but not all. Note that new error codes might be added later, and existing error codes might be updated. Your application should be prepared to do default handling for error scenarios.

Status code	Error type	Error description
400	INVALID_PARAMETERS	Some parameters in the request are invalid.
400	INVALID_WEBHOOK_KEY	Invalid webhook key.
400	MISSING_REQUIRED_PARAM	The required parameters are missing.
403	WEBHOOK_DELETION_NOT_ALLOWED	Webhook creation is not allowed.

See also the Format of error response section for details.