Nerddy

NOTIFICATIONS API

About the API

The notifications API, also known as output API, allows apps to send output, content, suggestions, warnings, and normal notifications to users.

Authentication

- Requires appid and accesskey.
- Some features require platform granted permission.

Limitations

- Notifications can’t include JavaScript unless granted platform permission.
- Each notification type should be used within the scope of its stated purposes.
- Links in notifications should open in a new tab. Users should not be redirected away from Nerddy at any point.
- Notifications may not include conflicting CSS. Notifications may not alter Nerddy’s design or make modifications to the layout.
- Notifications may not include PHP code.
- Notifications may not trick the users to install malware or send the user to malicious pages.
- Notifications may not be used to gather credit/debit card data unless granted a special permission in writing.
- Notifications may not contain advertisements in violation of Nerddy’s monetization guidelines.

API URL

http://www.nerddy.com/beta/notification/v1

Supported Methods

POST

Request Format

Here is a sample request body:

{"appid":"xxxx", "accesskey":"xxxx",
"requestid":"xxxx","design":"x","recipient":"xxxx","sessionid":"xxxx","queryid":"xxxx","title":"xxxx","tagline":"
xxxx","showsavebutton":"x","allowsharing":"x","allowpowerbutton":"x","notificationbody":"<contains-html>
Here could be any content or Nerddy mark-up </br> <iframe width="900" height="600"
src="https://www.youtube.com/embed/zz4bfDgtc8M" frameborder="0"></iframe> </contains-
html>"}

appid*	Your app's parent (global) app ID
accesskey*	Your parent app's access key
requestid*	A random integer (required for reference)
design*	Specify the design type of notification. Different types are explained in the types of notifications section. Design can be "output" or "content" or "sidebar" or "normal" or "bar"
recipient*	The ID of the user that the notification is sent to (varchar)
sessionid	The ID of the session on Nerddy where the user made the request. This could easily be accessed in the API communication stage if you specify #!usersessionid!# under return in input processing instructions. Sessionid is required for session saving to be enabled for your app. Session saving allows users to retrieve sessions they accessed earlier. If you wish to make an output notification irretrievable, don't specify a sessionid parameter.
queryid	Required for output type notifications. Specify the ID of the original user query. This could easily be accessed in the API communication stage if you specify #!userqueryid!# under return in input processing instructions.
title*	Required for all notification types. Title will be displayed to user in normal, sidebar, and bar notifications. For output and content notifications, a fixed title may be used or you may specify the original query.
tagline*	Required for all notification types. Tagline will be displayed to the user before the semicolon and title in normal and sidebar notifications. For normal and sidebar notifications, the recommended tagline is the title of your application. For bar type, the tagline is the text

	the user clicks to open the notification. For output and content notifications, use the title of your application as the tagline.
showsavebutton	Can be “yes” or “no”. If showsavebutton is yes, Nerddy will show a save button that allows the user to save the notification for future access.
allowsharing	Can be “yes” or “no”. If allowsharing is yes, Nerddy will show a sharing button that allows the user to create a share link that they could share with other users to share the notification content.
allowpowerbutton	Can be “yes” or “no”. If allowpowerbutton is yes, Nerddy will show a button that when clicked by the user, the content of the notification will automatically be shared to other Nerddy users who may be interested. These users may or may not have a connection with the original user who clicked the power button.
notificationbody*	This is the content of the notification. Content MUST be placed between <contains-html> </contains-html> tags even if it doesn’t contain HTML. Notification body may contain Nerddy mark-up which is presented in the Nerddy mark-up section in this manual. Body may not contain JavaScript unless permission is granted. It may contain HTML and CSS. No special encoding is required.

Response Format

The response will return the status of the request.

{"requestid":"xxxx","result":"success","error":null}

Types of Notifications

There are different types of notifications on Nerddy as shown below.

Type	**Uses and Limitations**
output	The content of the notification will be displayed directly to the user. A sample output notification is shown in Figure 1 in the appendix. This type of notification should only be used to return output to a user in response to a request or intent. Output notifications are not delivered to offline users.
content	Similar to output notifications, the content of the notification will be displayed directly to the user. Content notifications may be sent to offline users. Such notification type is ideal for delivering articles the user expressed interest in, notifications that are important for a limited time such as notification of completion of a file conversion or video processing, and reminders. Content notifications may not be sent to users who had never used your app and may not contain auto-playing video or GIFs. They may not be used to market or sell affiliate products.
normal	Normal notifications are normal notifications. However, on Nerddy, normal notifications are meant to be used for important information. Normal notifications are the user's "inbox" and may not be used to remind the user to use your app or buy your product.
sidebar	Sidebar notifications show under *Interesting* in the side menu and are

	meant to allow apps to reach new users and deliver content to users who may be interested. Apps may use the User Search API to find users who may be interested. You may not use sidebar notifications to send content to untargeted users. The content in sidebar notifications loads only when the notification is clicked. Apps that receive a relatively low click-to-notifications ratio will be flagged for review. You may not send “too many” notifications to the same users on one day.
bar	Bar notifications show up as a bar on the top of the page as shown in the example (Figure 2) in the appendix. Bar notifications may be used to send requests that require urgent action from the user when the user is known to be online. They may also be used to offer extra content such as offering hotel search results in addition to the requested flight search results.

Notification Body Markup

Nerddy markup allows apps to create interactive interfaces within notifications. There are many cases where you may need to use Nerddy markup in notifications. For instance, you may need to load pre-defined output to a user when an element such as some text or image is clicked. You may need to send an API request to a page on your website with dynamic payload when some text or image is clicked. Nerddy markup also allows your application to receive form data, create image albums, and paginate content.

Load new pre-defined content when a marked text/image is clicked. Marking tags : <NNB[i]></NNB[i]> #Use marking tags to define clickable text. Content tags: <NerddyNewBox[i]></NerddyNewBox[i]> #Use content tags to define content to be shown when marked text is clicked. Sub (nested content) marking tags: <NNB[i][b]></NNB[i][b]> #Use nested content marking tags to define clickable text that is contained in data under content tags. Sub content (nested content) tags: <NerddyNewBox[i][b]> </NerddyNewBox[i][b]> #Use nested content tags to define content to be shown when the associated nested content marking tag is clicked. • Only one level of nesting is supported. Content tags must be placed at the end of the level as shown in the implementation. • [i] must start with [1] and increase by increments of 1 for other boxes at the same level relative to parent. The same applies for [b]. http://www.example.com</DRURL[2][1]>	Implementation: "notificationbody":"<contains-html><NNB[1]>Click here</NNB[1]> for more information. <NNB[2]>Click here</NNB[2]> for photos. <NerddyNewBox[1]> French fries are delicious. To learn about French fries history <NNB[1][1]>click this</NNB[1][1]>. To view some recipes click <NNB[1][2]>here</NNB[1][2]>. <NerddyNewBox[1][1]>French fries history. //No more levels of nesting allowed.</NerddyNewBox[1][1]> <NerddyNewBox[1][2]>French Fries recipes. HTML. CSS. Nerddy Code.</NerddyNewBox[1][2]> </NerddyNewBox[1]> <NerddyNewBox[2]> <nrdalbum>http://www.example.com/1.jpg, http://www.example.com/2.jpg</nrdalbum> <DR[2][1]>Click here</DR[2][1]> to order <DataRequest[2][1]><DRURL[2][1]> {"key"="123456", "item"="12386", "type"="sides"}</DataRequest[2][1]> </NerddyNewBox[2]></contains-html>"

Send a request to third party server with defined payload when a text/image is clicked <u>Marking tags:</u> <DR[i]></DR[i]> #Use marking tags to define clickable text. <u>Payload tags:</u> Payload container tags: <DataRequest[i]></DataRequest[i]> API URL must be given under <DRURL[i]></DRURL[i]> tags inside the container before payload data is defined. Payload data must be in JSON. See implementation. When marked text/image is clicked, data will be sent to associated DRURL in this format: {"auth":"vericode":"xxxx","appid":"xxxx","requestid":"xxxx","userip":"x.x.x.x","userid":"xxxx"},"body":{"onion"="1234", "item"="12398","type"="roasted"}} Method: POST	Implementation: "notificationbody":"<contains-html><DR[1]>Click here</DR[1> to order a potato. <DR[2]>Click this</DR[2]> to order a roasted onion. <DataRequest[1]><DRURL[1]> http://www.example.com/api.php</DRURL[1]> {"item"="12386", "type"="cheesy"}</DataRequest[1]> <DataRequest[2]><DRURL[2]> http://www.example.com/abc.php</DRURL[2]> {"onion"="1234", "item"="12398", "type"="roasted"}</DataRequest[2]> </contains-html>"
Receive form data to an action URL Same as HTML. Simply specify your API's URL as the form action URL. Method: POST • Forms can be included under main content and under subcontent (NerddyNewBox). • Forms can be paginated using the pagination tags. <u>Format form data is sent to the action URL:</u>	Implementation: "notificationbody":"<contains-html>*<form action= "http://www.example.com/receiveformdata.php" > First name: <input type="text" name="firstname" value="Mickey"> Last name: <input type="text" name="lastname" value="Mouse"> <input type="submit" value="Submit"> </form></contains-html>*"

{"auth":{"vericode":"xxxx","appid":"xxxx","requestid":"xxxx","userip":"x.x.x.x","userid":"xxxx"}, "body":{"firstname":"Steve","lastname":"Jobs"}} Authentication information will always be included. Please verify that the vericode and app id are authentic. We further recommend that you verify that the requestid is valid for the userid.	
Display images in an image gallery Tags: <nrdalbum></nrdalbum> • Must separate image URLs using commas. • Images are paginated automatically. • Image galleries can be included under main content and under subcontent (NerddyNewBox).	Implementation: "notificationbody":"<contains-html> *<nrdalbum>http://www.example.com/image1.jpg , http://www.example.org/sample.png, http://www.example.net/cat.jpg</nrdalbum> </contains-html>*"
Paginate content Tags: <page[i]></page[i]> • Pagination tags can be included under main content or nested content. • *Pagination tags don't work with nrdalbum.*	Implementation: "notificationbody":"<contains-html> *<page[1]>Hello</page[1]><page[2]>Hi this is page 2 <img src=*"http://www.example.com"><*/page[2]> </contains-html>*"
Display user list Tags: <nrduser></nrduser> • This feature displays an interactive list of users. • Feature is only available to apps that use the Profiles API. Please read the Profiles API documentation first. • Content under onlistsummary for each user in the list will be shown. • Must separate userids using commas.	Implementation: "notificationbody":"<contains-html> *<nrduser>724607209309728,8279861767812,467268716782197,3816876729807817,1761892781926289827</nrduser> </contains-html>*"

• When a user is clicked, their Nerddy avatar, full name, username, and profilebody stored by your app for the user through the Profiles API will be displayed. • User list is automatically paginated. Ten users per page will be shown.	
Return the list of threads/conversations between the notification recipient and another defined user *Tags:* <nrdthreads> userID</nrdthreads> • Returns list of conversations between two users. When a conversation is clicked, the associated message room will load.	Implementation: "notificationbody":"<contains-html> *<nrdthreads>849750854795943865</nrdthreads>* *</contains-html>*" Where first user is the notification recipient and second user is *849750854795943865*
Show a new conversation button to allow the notification recipient to start a conversation with a defined user *Tags:* <nrdnewthread> userID</nrdnewthread> • Returns a "New Thread" button on the top right of the content container. When that button is clicked, the notification recipient will be asked to give the conversation a title and a new message room will be created for the notification recipient and the user defined under the notification tag.	Implementation: "notificationbody":"<contains-html> *<nrdnewthread>849750854795943865</nrdnewthread>* *</contains-html>*" Where 849750854795943865 is the ID of the defined user
Show a specific conversation by thread ID *Tags:* <nrdconversation> threadID</nrdconversation> • Returns an existing message room by thread ID. • Thread ID should be given between the tags.	Implementation: "notificationbody":"<contains-html> *<nrdconversation>5f6c57574585643634ff332478* *</nrdconversation >* *</contains-html>*"

Messages API

About the API

Messages API allows apps to start app-to-user conversations with a particular user. The most common use of the Messages API is asking the user for information they didn't provide in the query. Did you know that Nerddy offers a natural language processing API that is built especially for the messaging system?

Authentication

• Requires appid and accesskey.

• Requires platform granted permission.

Limitations

• This feature should NOT be used to advertise services to users without user initiation. The system is closely monitored.

API URL

http://www.nerddy.com/beta/messagesapi/v1

Supported Methods

POST

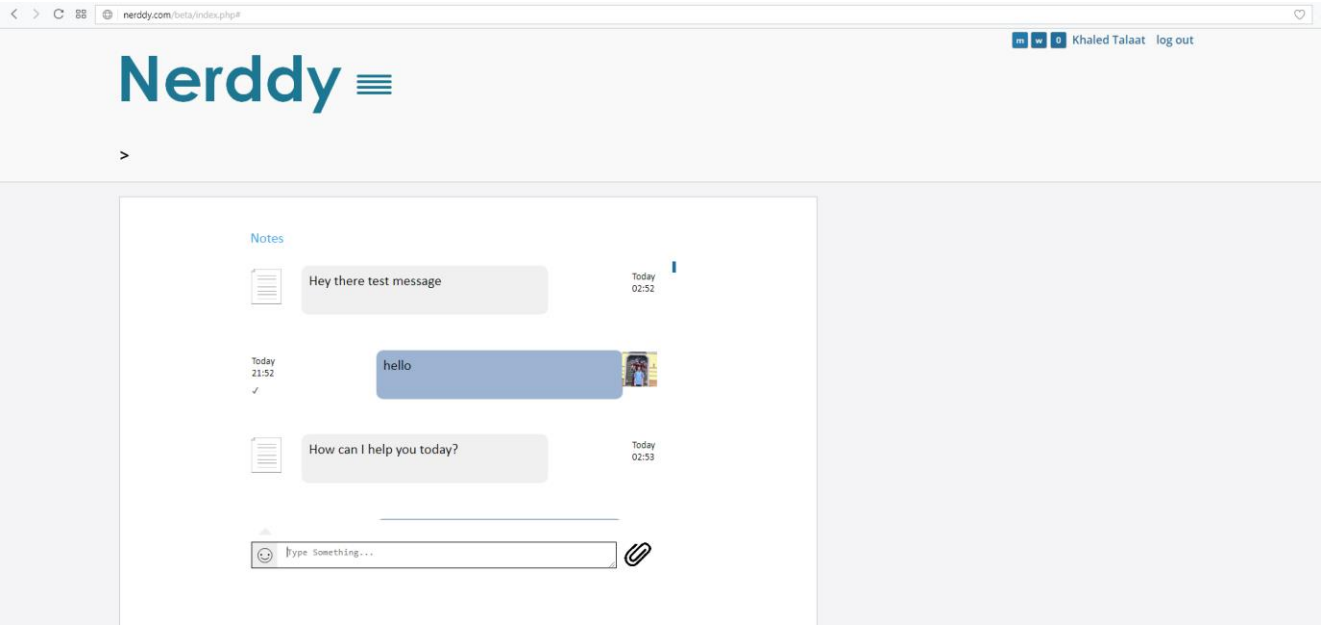
Request Format

Here is a sample request body:

{"appid":"xxxx","accesskey":"xxxx",
"requestid":"xxxx","userid":"xxxx","threadid":"xxxx","messageid":"xxxx","conversationtype":"x"
,"threadtitle":"xxxx","messagecontent":"xxxx","attachedfiles":"xxxx"}

appid	Your app's parent (global) app ID.
accesskey	Your parent app's access key.
requestid	A random integer (required for reference).
userid	The ID of the user (varchar).
threadid	Specify the ID of the

	thread/conversation/message room that the message should be sent to. This can be an existing room or a new one. The value should be a varchar longer than 20 characters.
messageid	A varchar ID for the message for reference. The value should be 20 characters or longer.
conversationtype	Integer. The only supported value currently is 2. If specified otherwise, you'll not be able to receive a response from the user.
threadtitle	Title of the conversation room. This will be shown to the user.
messagecontent	The content of your message. Currently, this can only be plain text. No HTML or Nerddy Markup.
attachedfiles	Comma-separated values to the URLs where the files could be directly accessed.



A Sample App-to-User Conversation

Important Notes:

• When you initiate a conversation with a user, Nerddy will not send the conversation room to the user. After you send the first message, you should use

the Notifications API to send the conversation to the user by thread ID. Don't do that every time you send a message to the user in the same conversation.

• Users can access the conversations at any time. You may need to set an expiry flag on your end for the thread if you wish to end the conversation. For some applications, it doesn't make sense to end the conversation. For example, if you are selling pizzas, it's a good idea to keep the conversation alive so that the user could return back to it at any time and re-order the pizza.

Receiving Messages from Users

To receive messages from users, you have to first set an end-point to which messages should be sent. To do this, please refer to the dev commands table in the developer's manual.

Nerddy will send user messages to your end-point in this format:

{"appid":"xxxx", "vericode":"xxxx","requestid":"xxxx", "userid":"xxxx","threadid":"xxxx","messageid":"xxxx","threadtitle":"xxxx","messagecontent":"xxxx","attachedfiles":"xxxx"}

vericode	The vericode generated for your app upon registration. Refer to the developer manual for how to know the vericode associated with your app.
appid	Your app's parent (global) app ID.
requestid	A random integer.
userid	The ID of the user that sent the message.
threadid	The ID of the thread/conversation room where the message was sent.
messageid	The ID of the message sent by the user on Nerddy (varchar).
threadtitle	Title of the conversation room.
messagecontent	The text content of the message sent by the user.
attachedfiles	Comma-separated values to the URLs where the files attached by the user could be directly accessed for 10 minutes.

We recommend that you use Nerddy NLP API to analyze the messagecontent sent to you by the user. Nerddy NLP recognizes many types of entities and can relate different messages.

Appendix

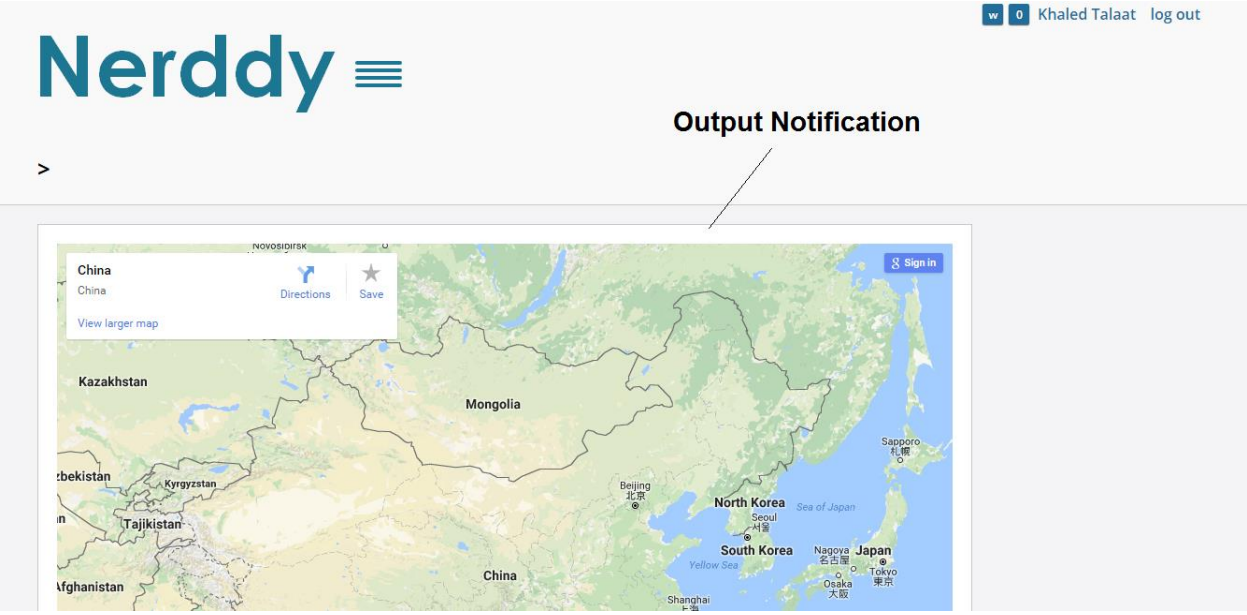


Figure 1: Sample Output Notification

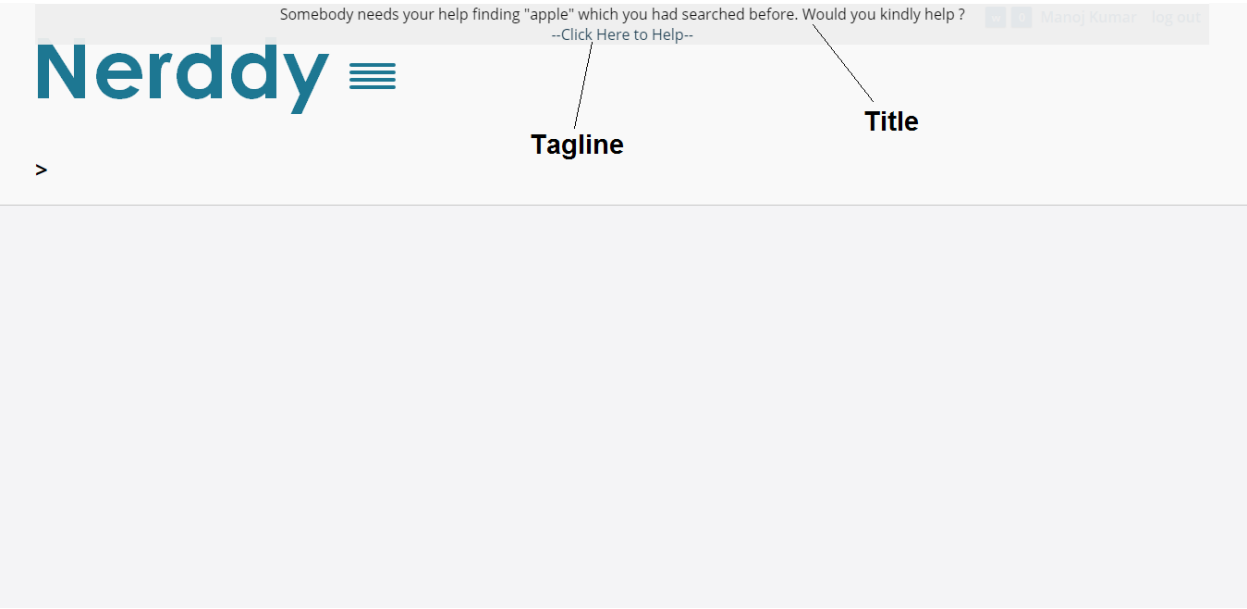


Figure 2: Sample Bar Notification