

Instalação Apprise

Link: <https://github.com/caronc/apprise#productivity-based-notifications>

git clone

<https://github.com/caronc/apprise.git>



ap·prise / *verb*

To inform or tell (someone). To make one aware of something.

Apprise allows you to send a notification to *almost* all of the most popular *notification* services available to us today such as: Telegram, Discord, Slack, Amazon SNS, Gotify, etc.

- One notification library to rule them all.
- A common and intuitive notification syntax.
- Supports the handling of images and attachments (*to the notification services that will accept them*).
- It's incredibly lightweight.
- Amazing response times because all messages sent asynchronously.

Developers who wish to provide a notification service no longer need to research each and every one out there. They no longer need to try to adapt to the new ones that come out thereafter. They just need to include this one library and then they can immediately gain access to almost all of the notifications services available to us today.

System Administrators and DevOps who wish to send a notification now no longer need to find the right tool for the job. Everything is already wrapped and supported within the `apprise` command line tool (CLI) that ships with this product.

Paypal Follow

Discord Python Build Status CodeCov Status PyPi Downloads

Table of Contents

- [Supported Notifications](#)
 - [Productivity Based Notifications](#)
 - [SMS Notifications](#)
 - [Desktop Notifications](#)
 - [Email Notifications](#)
 - [Custom Notifications](#)
- [Installation](#)
- [Command Line Usage](#)
 - [Configuration Files](#)
 - [File Attachments](#)
 - [Loading Custom Notifications/Hooks](#)
 - [Environment Variables](#)
- [Developer API Usage](#)
 - [Configuration Files](#)
 - [File Attachments](#)
 - [Loading Custom Notifications/Hooks](#)
- [Persistent Storage](#)
- [More Supported Links and Documentation](#)

Supported Notifications

The section identifies all of the services supported by this library. [Check out the wiki for more information on the supported modules here.](#)

Productivity Based Notifications

The table below identifies the services this tool supports and some example service urls you need to use in order to take advantage of it. Click on any of the services listed below to get more details on how you can configure Apprise to access them.

Notification Service	Service ID	Default Port	Example Syntax
Apprise API	apprise:// or apprises://	(TCP) 80 or 443	apprise://hostname/Token

Notification Service	Service ID	Default Port	Example Syntax
<u>AWS SES</u>	ses://	(TCP) 443	ses://user@domain/AccessKeyID/AccessSecretKey/RegionName ses://user@domain/AccessKeyID/AccessSecretKey/RegionName/email1/email2/emailN
<u>Bark</u>	bark://	(TCP) 80 or 443	bark://hostname bark://hostname/device_key bark://hostname/device_key1/device_key2/device_keyN barks://hostname barks://hostname/device_key barks://hostname/device_key1/device_key2/device_keyN
<u>BlueSky</u>	bluesky://	(TCP) 443	bluesky://Handle:AppPw bluesky://Handle:AppPw/TargetHandle bluesky://Handle:AppPw/TargetHandle1/TargetHandle2/TargetHandleN
<u>Chanify</u>	chantify://	(TCP) 443	chantify://token
<u>Discord</u>	discord://	(TCP) 443	discord://webhook_id/webhook_token discord://avatar@webhook_id/webhook_token
<u>Emby</u>	emby:// or embys://	(TCP) 8096	emby://user@hostname/ emby://user:password@hostname
<u>Enigma2</u>	enigma2:// or enigma2s://	(TCP) 80 or 443	enigma2://hostname
<u>FCM</u>	fcm://	(TCP) 443	fcm://project@apikey/DEVICE_ID fcm://project@apikey/#TOPIC fcm://project@apikey/DEVICE_ID1/#topic1/#topic2/DEVICE_ID2/
<u>Feishu</u>	feishu://	(TCP) 443	feishu://token

Notification Service	Service ID	Default Port	Example Syntax
<u>Flock</u>	flock://	(TCP) 443	flock://token flock://botname@token flock://app_token/u:userid flock://app_token/g:channel_id flock://app_token/u:userid/g:channel_id
<u>Google Chat</u>	gchat://	(TCP) 443	gchat://workspace/key/token
<u>Gotify</u>	gotify:// or gotifys://	(TCP) 80 or 443	gotify://hostname/token gotifys://hostname/token?priority=high
<u>Growl</u>	growl://	(UDP) 23053	growl://hostname growl://hostname:portno growl://password@hostname growl://password@hostname:port Note: you can also use the get parameter <i>version</i> which can allow the growl request to behave using the older v1.x protocol. An example would look like: growl://hostname?version=1
<u>Guided</u>	guided://	(TCP) 443	guided://webhook_id/webhook_token guided://avatar@webhook_id/webhook_token
<u>Home Assistant</u>	hassio:// or hassios://	(TCP) 8123 or 443	hassio://hostname/accesstoken hassio://user@hostname/accesstoken hassio://user:password@hostname:port/accesstoken hassio://hostname/optional/path/accesstoken
<u>IFTTT</u>	ifttt://	(TCP) 443	ifttt://webhooksID/Event ifttt://webhooksID/Event1/Event2/EventN ifttt://webhooksID/Event1/?+Key=Value ifttt://webhooksID/Event1/?-Key=value1

Notification Service	Service ID	Default Port	Example Syntax
<u>Join</u>	join://	(TCP) 443	join://apikey/device join://apikey/device1/device2/deviceN/ join://apikey/group join://apikey/groupA/groupB/groupN join://apikey/DeviceA/groupA/groupN/DeviceN/
<u>KODI</u>	kodi:// or kdis://	(TCP) 8080 or 443	kodi://hostname kodi://user@hostname kodi://user:password@hostname:port
<u>Kumulos</u>	kumulos://	(TCP) 443	kumulos://apikey/serverkey
<u>LaMetric Time</u>	lametric://	(TCP) 443	lametric://apikey@device_ip addr lametric://apikey@hostname:port lametric://client_id@client_secret
<u>Line</u>	line://	(TCP) 443	line://Token@User line://Token/User1/User2/UserN
<u>LunaSea</u>	lunasea://	(TCP) 80 or 443	lunasea://user:pass@+FirebaseDevice/ lunasea://user:pass@FirebaseUser/ lunasea://user:pass@hostname/+FirebaseDevice/ lunasea://user:pass@hostname/@FirebaseUser/
<u>Mailgun</u>	mailgun://	(TCP) 443	mailgun://user@hostname/apikey mailgun://user@hostname/apikey/email mailgun://user@hostname/apikey/email1/email2/emailN mailgun://user@hostname/apikey/?name="From%20User"
<u>Mastodon</u>	mastodon:// or mastodons://	(TCP) 80 or 443	mastodon://access_key@hostname mastodon://access_key@hostname/@user mastodon://access_key@hostname/@user1/@user2/@userN

Notification Service	Service ID	Default Port	Example Syntax
<u>Matrix</u>	matrix:// or matrixs://	(TCP) 80 or 443	matrix://hostname matrix://user@hostname matrixs://user:pass@hostna me:port/#room_alias matrixs://user:pass@hostna me:port/!room_id matrixs://user:pass@hostna me:port/#room_alias/!room _id/#room2 matrixs://token@hostname: port/?webhook=matrix matrix://user:token@hostna me/?webhook=slack&forma t=markdown
<u>Mattermost</u>	mmost:// or mmosts://	(TCP) 8065	mmost://hostname/authkey mmost://hostname:80/auth key mmost://user@hostname:8 0/authkey mmost://hostname/authkey ?channel=channel mmosts://hostname/authke y mmosts://user@hostname/a uthkey
<u>Microsoft Power Automate / Workflows (MSTeams)</u>	workflows://	(TCP) 443	workflows://WorkflowID/Sig nature/
<u>Microsoft Teams</u>	msteams://	(TCP) 443	msteams://TokenA/TokenB/ TokenC/
<u>Misskey</u>	misskey:// or misskeys://	(TCP) 80 or 443	misskey://access_token@ho stname
<u>MQTT</u>	mqtt:// or mqttts://	(TCP) 1883 or 8883	mqtt://hostname/topic mqtt://user@hostname/topi c mqttts://user:pass@hostnam e:9883/topic
<u>Nextcloud</u>	ncloud:// or nclouds://	(TCP) 80 or 443	ncloud://adminuser:pass@h ost/User nclouds://adminuser:pass@ host/User1/User2/UserN
<u>NextcloudTalk</u>	nctalk:// or nctalks://	(TCP) 80 or 443	nctalk://user:pass@host/Ro omId nctalks://user:pass@host/R oomId1/RoomId2/RoomIdN
<u>Notica</u>	notica://	(TCP) 443	notica://Token/

Notification Service	Service ID	Default Port	Example Syntax
<u>Notifiarr</u>	notifiarr://	(TCP) 443	notifiarr://apikey/#channel notifiarr://apikey/#channel1 /#channel2/#channelN
<u>Notifico</u>	notifico://	(TCP) 443	notifico://ProjectID/Message Hook/
<u>ntfy</u>	ntfy://	(TCP) 80 or 443	ntfy://topic/ ntfys://topic/
<u>Office 365</u>	o365://	(TCP) 443	o365://TenantID:AccountEm ail/ClientID/ClientSecret o365://TenantID:AccountEm ail/ClientID/ClientSecret/Tar getEmail o365://TenantID:AccountEm ail/ClientID/ClientSecret/Tar getEmail1/TargetEmail2/Tar getEmailN
<u>OneSignal</u>	onesignal://	(TCP) 443	onesignal://AppID@APIKey/ PlayerID onesignal://TemplateID:App ID@APIKey/UserID onesignal://AppID@APIKey/ #IncludeSegment onesignal://AppID@APIKey/ Email
<u>Opsgenie</u>	opsgenie://	(TCP) 443	opsgenie://APIKey opsgenie://APIKey/UserID opsgenie://APIKey/#Team opsgenie://APIKey/*Schedul e opsgenie://APIKey/^Escalati on
<u>PagerDuty</u>	pagerduty://	(TCP) 443	pagerduty://IntegrationKey @ApiKey pagerduty://IntegrationKey @ApiKey/Source/Componen t
<u>PagerTree</u>	pagertree://	(TCP) 443	pagertree://integration_id
<u>ParsePlatform</u>	parsep:// or parseps://	(TCP) 80 or 443	parsep://AppID:MasterKey@ Hostname parseps://AppID:MasterKey @Hostname

Notification Service	Service ID	Default Port	Example Syntax
<u>PopcornNotify</u>	popcorn://	(TCP) 443	popcorn:///ApiKey/ToPhoneNo popcorn:///ApiKey/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/ popcorn:///ApiKey/ToEmail popcorn:///ApiKey/ToEmail1/ToEmail2/ToEmailN/ popcorn:///ApiKey/ToPhoneNo1/ToEmail1/ToPhoneNoN/ToEmailN
<u>Prowl</u>	prowl://	(TCP) 443	prowl://apikey prowl://apikey/providerkey
<u>PushBullet</u>	pbul://	(TCP) 443	pbul://accesstoken pbul://accesstoken/#channel pbul://accesstoken/A_DEVICE_ID pbul://accesstoken/ <u>email@address.com</u> pbul://accesstoken/#channel/#channel2/ <u>email@address.net</u> /DEVICE
<u>Pushjet</u>	pjet:// or pjets://	(TCP) 80 or 443	pjet://hostname/secret pjet://hostname:port/secret pjets://secret@hostname/secret pjets://hostname:port/secret
<u>Push (Techulus)</u>	push://	(TCP) 443	push://apikey/
<u>Pushed</u>	pushed://	(TCP) 443	pushed://appkey/appsecret/ pushed://appkey/appsecret/#ChannelAlias pushed://appkey/appsecret/#ChannelAlias1/#ChannelAlias2/#ChannelAliasN pushed://appkey/appsecret/@UserPushedID pushed://appkey/appsecret/@UserPushedID1/@UserPushedID2/@UserPushedIDN
<u>PushMe</u>	pushme://	(TCP) 443	pushme://Token/

Notification Service	Service ID	Default Port	Example Syntax
<u>Pushover</u>	pover://	(TCP) 443	pover://user@token pover://user@token/DEVICE pover://user@token/DEVICE 1/DEVICE2/DEVICEN Note: you must specify both your user_id and token
<u>PushSafer</u>	psafer:// or psafers://	(TCP) 80 or 443	psafer://privatekey psafers://privatekey/DEVICE psafer://privatekey/DEVICE 1/DEVICE2/DEVICEN
<u>Pushy</u>	pushy://	(TCP) 443	pushy://apikey/DEVICE pushy://apikey/DEVICE1/DE VICE2/DEVICEN pushy://apikey/TOPIC pushy://apikey/TOPIC1/TOPI C2/TOPICN
<u>PushDeer</u>	pushdeer:// or pushdeers://	(TCP) 80 or 443	pushdeer://pushKey pushdeer://hostname/pushK ey pushdeer://hostname:port/p ushKey
<u>Reddit</u>	reddit://	(TCP) 443	reddit://user:password@app _id/app_secret/subreddit reddit://user:password@app _id/app_secret/sub1/sub2/s ubN
<u>Resend</u>	resend://	(TCP) 443	resend://APIToken:FromEm ail/ resend://APIToken:FromEm ail/ToEmail resend://APIToken:FromEm ail/ToEmail1/ToEmail2/ToE mailN/
<u>Revolt</u>	revolt://	(TCP) 443	revolt://bottoken/ChannelID revolt://bottoken/ChannelID 1/ChannelID2/ChannelIDN
<u>Rocket.Chat</u>	rocket:// or rockets://	(TCP) 80 or 443	rocket://user:password@ho stname/RoomID/Channel rockets://user:password@h ostname:443/#Channel1/# Channel1/RoomID rocket://user:password@ho stname/#Channel rocket://webhook@hostnam e rockets://webhook@hostna me/@User/#Channel

Notification Service	Service ID	Default Port	Example Syntax
<u>RSyslog</u>	rsyslog://	(UDP) 514	rsyslog://hostname rsyslog://hostname/Facility
<u>Ryver</u>	ryver://	(TCP) 443	ryver://Organization/Token ryver://botname@Organization/Token
<u>SendGrid</u>	sendgrid://	(TCP) 443	sendgrid://APIToken:FromEmail/ sendgrid://APIToken:FromEmail/ToEmail sendgrid://APIToken:FromEmail/ToEmail1/ToEmail2/ToEmailN/
<u>ServerChan</u>	schan://	(TCP) 443	schan://sendkey/
<u>Signal API</u>	signal:// or signals://	(TCP) 80 or 443	signal://hostname:port/FromPhoneNo signal://hostname:port/FromPhoneNo/ToPhoneNo signal://hostname:port/FromPhoneNo/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/
<u>SimplePush</u>	spush://	(TCP) 443	spush://apikey spush://salt:password@apikey spush://apikey?event=Apprise
<u>Slack</u>	slack://	(TCP) 443	slack://TokenA/TokenB/TokenC/ slack://TokenA/TokenB/TokenC/Channel slack://botname@TokenA/TokenB/TokenC/Channel slack://user@TokenA/TokenB/TokenC/Channel1/Channel2/ChannelN
<u>SMTP2Go</u>	smtp2go://	(TCP) 443	smtp2go://user@hostname/apikey smtp2go://user@hostname/apikey/email smtp2go://user@hostname/apikey/email1/email2/emailN smtp2go://user@hostname/apikey/?name="From%20User"

Notification Service	Service ID	Default Port	Example Syntax
<u>SparkPost</u>	sparkpost://	(TCP) 443	sparkpost://user@hostname /apikey sparkpost://user@hostname /apikey/email sparkpost://user@hostname /apikey/email1/email2/emai IN sparkpost://user@hostname /apikey/?name="From%20 User"
<u>Splunk</u>	splunk:// or victorops:/	(TCP) 443	splunk://route_key@apikey splunk://route_key@apikey/ entity_id
<u>Streamlabs</u>	strmlabs://	(TCP) 443	strmlabs://AccessToken/ strmlabs://AccessToken/?na me=name&identifier=ident ifier&amount=0¤cy= USD
<u>Synology Chat</u>	synology:// or synologys://	(TCP) 80 or 443	synology://hostname/token synology://hostname:port/t oken
<u>Syslog</u>	syslog://	n/a	syslog:// syslog://Facility
<u>Telegram</u>	tgram://	(TCP) 443	tgram://bottoken/ChatID tgram://bottoken/ChatID1/C hatID2/ChatIDN
<u>Twitter</u>	twitter://	(TCP) 443	twitter://CKey/CSecret/AKey /ASecret twitter://user@CKey/CSecre t/AKey/ASecret twitter://CKey/CSecret/AKey /ASecret/User1/User2/User2 twitter://CKey/CSecret/AKey /ASecret?mode=tweet
<u>Twist</u>	twist://	(TCP) 443	twist://password:login twist://password:login/#cha nnel twist://password:login/#tea m:channel twist://password:login/#tea m:channel1/channel2/#tea m3:channel
<u>Webex Teams (Cisco)</u>	wxteams://	(TCP) 443	wxteams://Token
<u>WeCom Bot</u>	wecombot://	(TCP) 443	wecombot://BotKey

Notification Service	Service ID	Default Port	Example Syntax
<u>WhatsApp</u>	whatsapp://	(TCP) 443	whatsapp://AccessToken@FromPhoneID/ToPhoneNo whatsapp://Template:AccessToken@FromPhoneID/ToPhoneNo
<u>WxPusher</u>	wxpusher://	(TCP) 443	wxpusher://AppToken@UserID1/UserID2/UserIDN wxpusher://AppToken@Topic1/Topic2/Topic3 wxpusher://AppToken@UserID1/Topic1/
<u>XBMC</u>	xbmc:// or xbmc://	(TCP) 8080 or 443	xbmc://hostname xbmc://user@hostname xbmc://user:password@hostname:port
<u>Zulip Chat</u>	zulip://	(TCP) 443	zulip://botname@Organization/Token zulip://botname@Organization/Token/Stream zulip://botname@Organization/Token/Email

SMS Notifications

SMS Notifications for the most part do not have a both a `title` and `body`. They consist of a single `body` which is usually no more then 160 characters in length. When using Apprise, the `title` and `body` are therefore combined into a single message prior to their transmission.

Notification Service	Service ID	Default Port	Example Syntax
<u>Africas Talking</u>	atalk://	(TCP) 443	atalk://AppUser@ApiKey/ToPhoneNo atalk://AppUser@ApiKey/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/
<u>Automated Packet Reporting System (ARPS)</u>	aprs://	(TCP) 10152	aprs://user:pass@callsign aprs://user:pass@callsign1/callsign2/callsignN

Notification Service	Service ID	Default Port	Example Syntax
<u>AWS SNS</u>	sns://	(TCP) 443	sns://AccessKeyID/AccessSecretKey/RegionName/+PhoneNo sns://AccessKeyID/AccessSecretKey/RegionName/+PhoneNo1/+PhoneNo2/+PhoneNoN sns://AccessKeyID/AccessSecretKey/RegionName/Topic sns://AccessKeyID/AccessSecretKey/RegionName/Topic1/Topic2/TopicN
<u>BulkSMS</u>	bulksms://	(TCP) 443	bulksms://user:password@ToPhoneNo bulksms://User:Password@ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/
<u>BulkVS</u>	bulkvs://	(TCP) 443	bulkvs://user:password@FromPhoneNo bulkvs://user:password@FromPhoneNo/ToPhoneNo bulkvs://user:password@FromPhoneNo/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/
<u>Burst SMS</u>	burstsms://	(TCP) 443	burstsms://ApiKey:ApiSecret@FromPhoneNo/ToPhoneNo burstsms://ApiKey:ApiSecret@FromPhoneNo/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/
<u>ClickSend</u>	clicksend://	(TCP) 443	clicksend://user:pass@PhoneNo clicksend://user:pass@ToPhoneNo1/ToPhoneNo2/ToPhoneNoN
<u>DAPNET</u>	dapnet://	(TCP) 80	dapnet://user:pass@callsign dapnet://user:pass@callsign1/callsign2/callsignN
<u>D7 Networks</u>	d7sms://	(TCP) 443	d7sms://token@PhoneNo d7sms://token@ToPhoneNo1/ToPhoneNo2/ToPhoneNoN
<u>DingTalk</u>	dingtalk://	(TCP) 443	dingtalk://token/ dingtalk://token/ToPhoneNo dingtalk://token/ToPhoneNo1/ToPhoneNo2/ToPhoneNo1/ /

Notification Service	Service ID	Default Port	Example Syntax
<u>Free-Mobile</u>	freemobile://	(TCP) 443	freemobile://user@password/
<u>httpSMS</u>	httpsms://	(TCP) 443	httpsms://ApiKey@FromPhoneNo httpsms://ApiKey@FromPhoneNo/ToPhoneNo httpsms://ApiKey@FromPhoneNo/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/
<u>Kavenegar</u>	kavenegar://	(TCP) 443	kavenegar://ApiKey/ToPhoneNo kavenegar://FromPhoneNo@ApiKey/ToPhoneNo kavenegar://ApiKey/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN
<u>MessageBird</u>	msgbird://	(TCP) 443	msgbird://ApiKey/FromPhoneNo msgbird://ApiKey/FromPhoneNo/ToPhoneNo msgbird://ApiKey/FromPhoneNo/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/
<u>MSG91</u>	msg91://	(TCP) 443	msg91://TemplateID@AuthKey/ToPhoneNo msg91://TemplateID@AuthKey/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/
<u>Plivo</u>	plivo://	(TCP) 443	plivo://AuthID@Token@FromPhoneNo plivo://AuthID@Token/FromPhoneNo/ToPhoneNo plivo://AuthID@Token/FromPhoneNo/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/
<u>Seven</u>	seven://	(TCP) 443	seven://ApiKey/FromPhoneNo seven://ApiKey/FromPhoneNo/ToPhoneNo seven://ApiKey/FromPhoneNo/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/
<u>Société Française du Radiotéléphone (SFR)</u>	sfr://	(TCP) 443	sfr://user:password>@spaceId/ToPhoneNo sfr://user:password>@spaceId/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/

Notification Service	Service ID	Default Port	Example Syntax
<u>Signal API</u>	signal:// or signals://	(TCP) 80 or 443	signal://hostname:port/FromPhoneNo signal://hostname:port/FromPhoneNo/ToPhoneNo signal://hostname:port/FromPhoneNo/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/
<u>Sinch</u>	sinch://	(TCP) 443	sinch://ServicePlanId:ApiToken@FromPhoneNo sinch://ServicePlanId:ApiToken@FromPhoneNo/ToPhoneNo sinch://ServicePlanId:ApiToken@FromPhoneNo/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/ sinch://ServicePlanId:ApiToken@ShortCode/ToPhoneNo sinch://ServicePlanId:ApiToken@ShortCode/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/
<u>SMSEagle</u>	smseagle:// or smseagles://	(TCP) 80 or 443	smseagles://hostname:port/ToPhoneNo smseagles://hostname:port/@ToContact smseagles://hostname:port/#ToGroup smseagles://hostname:port/ToPhoneNo1/#ToGroup/@ToContact/
<u>SMS Manager</u>	smsmgr://	(TCP) 443	smsmgr://ApiKey@ToPhoneNo smsmgr://ApiKey@ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/
<u>Threema Gateway</u>	threema://	(TCP) 443	threema://GatewayID@secret/ToPhoneNo threema://GatewayID@secret/ToEmail threema://GatewayID@secret/ToThreemaID/ threema://GatewayID@secret/ToEmail/ToThreemaID/ToPhoneNo/...

Notification Service	Service ID	Default Port	Example Syntax
<u>Twilio</u>	twilio://	(TCP) 443	twilio://AccountSid:AuthToken@FromPhoneNo twilio://AccountSid:AuthToken@FromPhoneNo/ToPhoneNo twilio://AccountSid:AuthToken@FromPhoneNo/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/ twilio://AccountSid:AuthToken@FromPhoneNo/ToPhoneNo?apikey=Key twilio://AccountSid:AuthToken@ShortCode/ToPhoneNo twilio://AccountSid:AuthToken@ShortCode/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/
<u>Voipms</u>	voipms://	(TCP) 443	voipms://password:email/FromPhoneNo voipms://password:email/FromPhoneNo/ToPhoneNo voipms://password:email/FromPhoneNo/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/
<u>Vonage</u> (formerly Nexmo)	nexmo://	(TCP) 443	nexmo://ApiKey:ApiSecret@FromPhoneNo nexmo://ApiKey:ApiSecret@FromPhoneNo/ToPhoneNo nexmo://ApiKey:ApiSecret@FromPhoneNo/ToPhoneNo1/ToPhoneNo2/ToPhoneNoN/

Desktop Notifications

Notification Service	Service ID	Default Port	Example Syntax
<u>Linux DBus Notifications</u>	dbus:// qt:// glib:// kde://	n/a	dbus:// qt:// glib:// kde://
<u>Linux Gnome Notifications</u>	gnome://	n/a	gnome://
<u>MacOS X Notifications</u>	macosx://	n/a	macosx://
<u>Windows Notifications</u>	windows://	n/a	windows://

Email Notifications

Service ID	Default Port	Example Syntax
<u>mailto://</u>	(TCP) 25	mailto://userid:pass@domain.com mailto://domain.com?user=userid&pass=password mailto://domain.com:2525?user=userid&pass=password mailto:// <u>user@gmail.com</u> &pass=password mailto://mySendingUsername: <u>mySendingPassword@example.com</u> ?to= <u>receivingAddress@example.com</u> mailto://userid: <u>password@example.com</u> ?smtp=mail.example.com&from= <u>noreply@example.com</u> &name=no%20reply
<u>mailto://</u>	(TCP) 587	mailto://userid:pass@domain.com mailto://domain.com?user=userid&pass=password mailto://domain.com:465?user=userid&pass=password mailto:// <u>user@hotmail.com</u> &pass=password mailto://mySendingUsername: <u>mySendingPassword@example.com</u> ?to= <u>receivingAddress@example.com</u> mailto://userid: <u>password@example.com</u> ?smtp=mail.example.com&from= <u>noreply@example.com</u> &name=no%20reply

Apprise have some email services built right into it (such as yahoo, fastmail, hotmail, gmail, etc) that greatly simplify the mailto:// service. See more details [here](#).

Custom Notifications

Post Method	Service ID	Default Port	Example Syntax
-------------	------------	--------------	----------------

<u>Form</u>	form:// or forms://	(TCP) 80 or 443	form://hostname form://user@hostname form://user:password@hostname:port form://hostname/a/path/to/post/to
<u>JSON</u>	json:// or jsons://	(TCP) 80 or 443	json://hostname json://user@hostname json://user:password@hostname:port json://hostname/a/path/to/post/to
<u>XML</u>	xml:// or xmls://	(TCP) 80 or 443	xml://hostname xml://user@hostname xml://user:password@hostname:port xml://hostname/a/path/to/post/to

Installation

The easiest way to install this package is from pypi:

```
pip install apprise
```

Apprise is also packaged as an RPM and available through [EPEL](#) supporting CentOS, Redhat, Rocky, Oracle Linux, etc.

```
# Follow instructions on https://docs.fedoraproject.org/en-US/epel
# to get your system connected up to EPEL and then:
# Redhat/CentOS 7.x users
yum install apprise

# Redhat/CentOS 8.x+ and/or Fedora Users
dnf install apprise
```

You can also check out the [Graphical version of Apprise](#) to centralize your configuration and notifications through a manageable webpage.

Command Line Usage

A small command line interface (CLI) tool is also provided with this package called *apprise*. If you know the server urls you wish to notify, you can simply provide them all on the command line and send your notifications that way:

```
# Send a notification to as many servers as you want
# as you can easily chain one after another (the -vv provides some
# additional verbosity to help let you know what is going on):
apprise -vv -t 'my title' -b 'my notification body' \
  'mailto://myemail:mypass@gmail.com' \
  'pbul://o.gn5kj6nfhv736l7jC3cj3QLRiyhgl98b'

# If you don't specify a --body (-b) then stdin is used allowing
# you to use the tool as part of your every day administration:
cat /proc/cpuinfo | apprise -vv -t 'cpu info' \
  'mailto://myemail:mypass@gmail.com'

# The title field is totally optional
uptime | apprise -vv \
  'discord:///4174216298/JHMHl8qBe7bk2ZwO5U711o3dV_js'
```

CLI Configuration Files

No one wants to put their credentials out for everyone to see on the command line. No problem *apprise* also supports configuration files. It can handle both a specific [YAML format](#) or a very simple [TEXT format](#). You can also pull these configuration files via an HTTP query too! You can read more about the expected structure of the configuration files [here](#).

```
# By default if no url or configuration is specified apprise will attempt to load
# configuration files (if present) from:
# ~/.apprise
# ~/.apprise.yaml
# ~/.config/apprise.conf
# ~/.config/apprise.yaml
# /etc/apprise.conf
# /etc/apprise.yaml

# Also a subdirectory handling allows you to leverage plugins
# ~/.apprise/apprise
# ~/.apprise/apprise.yaml
# ~/.config/apprise/apprise.conf
# ~/.config/apprise/apprise.yaml
# /etc/apprise/apprise.yaml
# /etc/apprise/apprise.conf

# Windows users can store their default configuration files here:
```

```
# %APPDATA%/Apprise/apprise.conf
# %APPDATA%/Apprise/apprise.yaml
# %LOCALAPPDATA%/Apprise/apprise.conf
# %LOCALAPPDATA%/Apprise/apprise.yaml
# %ALLUSERSPROFILE%\Apprise\apprise.conf
# %ALLUSERSPROFILE%\Apprise\apprise.yaml
# %PROGRAMFILES%\Apprise\apprise.conf
# %PROGRAMFILES%\Apprise\apprise.yaml
# %COMMONPROGRAMFILES%\Apprise\apprise.conf
# %COMMONPROGRAMFILES%\Apprise\apprise.yaml

# The configuration files specified above can also be identified with a `.yaml`
# extension or even just entirely removing the `.conf` extension altogether.

# If you loaded one of those files, your command line gets really easy:
apprise -vv -t 'my title' -b 'my notification body'

# If you want to deviate from the default paths or specify more than one,
# just specify them using the --config switch:
apprise -vv -t 'my title' -b 'my notification body' \
  --config=/path/to/my/config.yaml

# Got lots of configuration locations? No problem, you can specify them all:
# Apprise can even fetch the configuration from over a network!
apprise -vv -t 'my title' -b 'my notification body' \
  --config=/path/to/my/config.yaml \
  --config=https://localhost/my/apprise/config
```

CLI File Attachments

Apprise also supports file attachments too! Specify as many attachments to a notification as you want.

```
# Send a funny image you found on the internet to a colleague:
apprise -vv --title 'Agile Joke' \
  --body 'Did you see this one yet?' \
  --attach https://i.redd.it/my2t4d2fx0u31.jpg \
  'mailto://myemail:mypass@gmail.com'

# Easily send an update from a critical server to your dev team
apprise -vv --title 'system crash' \
  --body 'I do not think Jim fixed the bug; see attached...' \
  --attach /var/log/myprogram.log \
  --attach /var/debug/core.2345 \
  --tag devteam
```

CLI Loading Custom Notifications/Hooks

To create your own custom `schema://` hook so that you can trigger your own custom code, simply include the `@notify` decorator to wrap your function.

```
from apprise.decorators import notify
#
# The below assumes you want to catch foobar:// calls:
#
@notify(on="foobar", name="My Custom Foobar Plugin")
def my_custom_notification_wrapper(body, title, notify_type, *args, **kwargs):
    """My custom notification function that triggers on all foobar:// calls
    """
    # Write all of your code here... as an example...
    print("{}: {} - {}".format(notify_type.upper(), title, body))

    # Returning True/False is a way to relay your status back to Apprise.
    # Returning nothing (None by default) is always interpreted as a Success
```

Once you've defined your custom hook, you just need to tell Apprise where it is at runtime.

```
# By default if no plugin path is specified apprise will attempt to load
# all plugin files (if present) from the following directory paths:
# ~/.apprise/plugins
# ~/.config/apprise/plugins
# /var/lib/apprise/plugins

# Windows users can store their default plugin files in these directories:
# %APPDATA%/Apprise/plugins
# %LOCALAPPDATA%/Apprise/plugins
# %ALLUSERSPROFILE%\Apprise\plugins
# %PROGRAMFILES%\Apprise\plugins
# %COMMONPROGRAMFILES%\Apprise\plugins

# If you placed your plugin file within one of the directories already defined
# above, then your call simply needs to look like:
apprise -vv --title 'custom override' \
    --body 'the body of my message' \
    foobar:\\

# However you can over-ride the path like so
apprise -vv --title 'custom override' \
```

```
--body 'the body of my message' \  
--plugin-path /path/to/my/plugin.py \  
foobar:\\
```

You can read more about creating your own custom notifications and/or hooks [here](#).

CLI Environment Variables

Those using the Command Line Interface (CLI) can also leverage environment variables to pre-set the default settings:

Variable	Description
<code>APPRISE_URLS</code>	Specify the default URLs to notify IF none are otherwise specified on the command line explicitly. If the <code>--config</code> (<code>-c</code>) is specified, then this will over-rides any reference to this variable. Use white space and/or a comma (,) to delimit multiple entries.
<code>APPRISE_CONFIG_PATH</code>	Explicitly specify the config search path to use (over-riding the default). The path(s) defined here must point to the absolute filename to open/reference. Use a semi-colon (;), line-feed (<code>\n</code>), and/or carriage return (<code>\r</code>) to delimit multiple entries.
<code>APPRISE_PLUGIN_PATH</code>	Explicitly specify the custom plugin search path to use (over-riding the default). Use a semi-colon (;), line-feed (<code>\n</code>), and/or carriage return (<code>\r</code>) to delimit multiple entries.
<code>APPRISE_STORAGE_PATH</code>	Explicitly specify the persistent storage path to use (over-riding the default).

Developer API Usage

To send a notification from within your python application, just do the following:

```
import apprise  
  
# Create an Apprise instance  
apobj = apprise.Apprise()  
  
# Add all of the notification services by their server url.  
# A sample email notification:  
apobj.add('mailto://myuserid:mypass@gmail.com')
```

```
# A sample pushbullet notification
apobj.add('pbul://o.gn5kj6nfhv736l7jC3cj3QLRiyhgI98b')

# Then notify these services any time you desire. The below would
# notify all of the services loaded into our Apprise object.
apobj.notify(
    body='what a great notification service!',
    title='my notification title',
)
```

API Configuration Files

Developers need access to configuration files too. The good news is their use just involves declaring another object (called *AppriseConfig*) that the *Apprise* object can ingest. You can also freely mix and match config and notification entries as often as you wish! You can read more about the expected structure of the configuration files [here](#).

```
import apprise

# Create an Apprise instance
apobj = apprise.Apprise()

# Create an Config instance
config = apprise.AppriseConfig()

# Add a configuration source:
config.add('/path/to/my/config.yml')

# Add another...
config.add('https://myserver:8080/path/to/config')

# Make sure to add our config into our apprise object
apobj.add(config)

# You can mix and match; add an entry directly if you want too
# In this entry we associate the 'admin' tag with our notification
apobj.add('mailto://myuser:mypass@hotmail.com', tag='admin')

# Then notify these services any time you desire. The below would
# notify all of the services that have not been bound to any specific
# tag.
apobj.notify(
    body='what a great notification service!',
    title='my notification title',
)
```

```
# Tagging allows you to specifically target only specific notification
# services you've loaded:
apobj.notify(
    body='send a notification to our admin group',
    title='Attention Admins',
    # notify any services tagged with the 'admin' tag
    tag='admin',
)

# If you want to notify absolutely everything (regardless of whether
# it's been tagged or not), just use the reserved tag of 'all':
apobj.notify(
    body='send a notification to our admin group',
    title='Attention Admins',
    # notify absolutely everything loaded, regardless on whether
    # it has a tag associated with it or not:
    tag='all',
)
```

API File Attachments

Attachments are very easy to send using the Apprise API:

```
import apprise

# Create an Apprise instance
apobj = apprise.Apprise()

# Add at least one service you want to notify
apobj.add('mailto://myuser:mypass@hotmail.com')

# Then send your attachment.
apobj.notify(
    title='A great photo of our family',
    body='The flash caused Jane to close her eyes! hah! :)',
    attach='/local/path/to/my/DSC_003.jpg',
)

# Send a web based attachment too! In the below example, we connect to a home
# security camera and send a live image to an email. By default remote web
# content is cached, but for a security camera we might want to call notify
# again later in our code, so we want our last image retrieved to expire(in
# this case after 3 seconds).
apobj.notify(
    title='Latest security image',
    attach='http://admin:password@hikvision-cam01/ISAPI/Streaming/channels/101/picture?cache=3'
)
```


To send more than one attachment, just use a list, set, or tuple instead:

```
import apprise

# Create an Apprise instance
apobj = apprise.Apprise()

# Add at least one service you want to notify
apobj.add('mailto://myuser:mypass@hotmail.com')

# Now add all of the entries we're interested in:
attach = (
    # ?name= allows us to rename the actual jpeg as found on the site
    # to be another name when sent to our recipient(s)
    'https://i.redd.it/my2t4d2fx0u31.jpg?name=FlyingToMars.jpg',

    # Now add another:
    '/path/to/funny/joke.gif',
)

# Send your multiple attachments with a single notify call:
apobj.notify(
    title='Some good jokes.',
    body='Hey guys, check out these!',
    attach=attach,
)
```

API Loading Custom Notifications/Hooks

By default, no custom plugins are loaded at all for those building from within the Apprise API. It's at the developers discretion to load custom modules. But should you choose to do so, it's as easy as including the path reference in the `AppriseAsset()` object prior to the initialization of your `Apprise()` instance.

For example:

```
from apprise import Apprise
from apprise import AppriseAsset

# Prepare your Asset object so that you can enable the custom plugins to
# be loaded for your instance of Apprise...
```

```

asset = AppriseAsset(plugin_paths="/path/to/scan")

# OR You can also generate scan more then one file too:
asset = AppriseAsset(
    plugin_paths=[
        # Iterate over all python libraries found in the root of the
        # specified path. This is NOT a recursive (directory) scan; only
        # the first level is parsed. HOWEVER, if a directory containing
        # an __init__.py is found, it will be included in the load.
        "/dir/containing/many/python/libraries",

        # An absolute path to a plugin.py to exclusively load
        "/path/to/plugin.py",

        # if you point to a directory that has an __init__.py file found in
        # it, then only that file is loaded (it's similar to point to a
        # absolute .py file. Hence, there is no (level 1) scanning at all
        # within the directory specified.
        "/path/to/dir/library"
    ]
)

# Now that we've got our asset, we just work with our Apprise object as we
# normally do
aobj = Apprise(asset=asset)

# If our new custom `foobar://` library was loaded (presuming we prepared
# one like in the examples above). then you would be able to safely add it
# into Apprise at this point
aobj.add('foobar://')

# Send our notification out through our foobar://
aobj.notify("test")

```

You can read more about creating your own custom notifications and/or hooks [here](#).

Persistent Storage

Persistent storage allows Apprise to cache re-occurring actions optionally to disk. This can greatly reduce the overhead used to send a notification.

There are 3 Persistent Storage operational states Apprise can operate using:

1. `auto`: Flush gathered cache information to the filesystem on demand. This option is incredibly light weight. This is the default behavior for all CLI usage.

- Developers who choose to use this operational mode can also force cached information manually if they choose.
 - The CLI will use this operational mode by default.
2. `flush`: Flushes any cache information to the filesystem during every transaction.
 3. `memory`: Effectively disable Persistent Storage. Any caching of data required by each plugin used is done in memory. Apprise effectively operates as it always did before persistent storage was available. This setting ensures no content is ever written to disk.
 - By default this is the mode Apprise will operate under for those developing with it unless they configure it to otherwise operate as `auto` or `flush`. This is done through the `AppriseAsset()` object and is explained further on in this documentation.

CLI Persistent Storage Commands

You can provide the keyword `storage` on your CLI call to see the persistent storage options available to you.

```
# List all of the occupied space used by Apprise's Persistent Storage:
apprise storage list
```

```
# list is the default option, so the following does the same thing:
apprise storage
```

```
# You can prune all of your storage older then 30 days
# and not accessed for this period like so:
apprise storage prune
```

```
# You can do a hard reset (and wipe all persistent storage) with:
apprise storage clean
```

You can also filter your results by adding tags and/or URL Identifiers. When you get a listing (`apprise storage list`), you may see:

```
# example output of 'apprise storage list':
```

```
1. f7077a65                0.00B  unused
   - matrixs://abcdef:****@synapse.example12.com/%23general?image=no&mode=off&version=3&msgtype...
   tags: team

2. 0e873a46                81.10B  active
   - tgram://W...U//?image=False&detect=yes&silent=no&preview=no&content=before&mdv=v1&format=m...
   tags: personal

3. abcd123                 12.00B  stale
```

The (persistent storage) cache states are:

- `unused`: This plugin has not committed anything to disk for reuse/cache purposes
- `active`: This plugin has written content to disk. Or at the very least, it has prepared a persistent storage location it can write into.
- `stale`: The system detected a location where a URL may have possibly written to in the past, but there is nothing linking to it using the URLs provided. It is likely wasting space or is no longer of any use.

You can use this information to filter your results by specifying *URL ID* (UID) values after your command. For example:

```
# The below commands continue with the example already identified above
# the following would match abcd123 (even though just ab was provided)
# The output would only list the 'stale' entry above
apprise storage list ab

# knowing our filter is safe, we could remove it
# the below command would not obstruct our other to URLs and would only
# remove our stale one:
apprise storage clean ab

# Entries can be filtered by tag as well:
apprise storage list --tag=team

# You can match on multiple URL ID's as well:
# The followin would actually match the URL ID's of 1. and .2 above
apprise storage list f 0
```

When using the CLI, Persistent storage is set to the operational mode of `auto` by default, you can change this by providing `--storage-mode=` (`-SM`) during your calls. If you want to ensure it's always set to a value of your choice.

For more information on persistent storage, [visit here](#).

API Persistent Storage Commands

For developers, persistent storage is set in the operational mode of `memory` by default.

It's at the developers discretion to enable it (by switching it to either `auto` or `flush`). Should you choose to do so: it's as easy as including the information in the `AppriseAsset()` object prior to the

initialization of your `Apprise()` instance.

For example:

```
from apprise import Apprise
from apprise import AppriseAsset
from apprise import PersistentStoreMode

# Prepare a location the persistent storage can write it's cached content to.
# By setting this path, this immediately assumes you wish to operate the
# persistent storage in the operational 'auto' mode
asset = AppriseAsset(storage_path="/path/to/save/data")

# If you want to be more explicit and set more options, then you may do the
# following
asset = AppriseAsset(
    # Set our storage path directory (minimum requirement to enable it)
    storage_path="/path/to/save/data",

    # Set the mode... the options are:
    # 1. PersistentStoreMode.MEMORY
    #    - disable persistent storage from writing to disk
    # 2. PersistentStoreMode.AUTO
    #    - write to disk on demand
    # 3. PersistentStoreMode.FLUSH
    #    - write to disk always and often
    storage_mode=PersistentStoreMode.FLUSH

    # The URL IDs are by default 8 characters in length. You can increase and
    # decrease it's value here. The value must be > 2. The default value is 8
    # if not otherwise specified
    storage_idlen=8,
)

# Now that we've got our asset, we just work with our Apprise object as we
# normally do
aobj = Apprise(asset=asset)
```

For more information on persistent storage, [visit here](#).

Want To Learn More?

If you're interested in reading more about this and other methods on how to customize your own notifications, please check out the following links:

- [📖 Using the CLI](#)

- [!\[\]\(2dc8cdc0c918df88cde61039ecf68682_img.jpg\) Development API](#)
- [!\[\]\(793119bf0d613bd9b598fb8668922511_img.jpg\) Troubleshooting](#)
- [!\[\]\(0a4819029e810ca9d2aba79260b63a4d_img.jpg\) Configuration File Help](#)
- [!\[\]\(5b78a2fafd05db5e14d20573d68ef9b3_img.jpg\) Create Your Own Custom Notifications](#)
- [!\[\]\(25fe2c0d7244c22c84de6bda963b471d_img.jpg\) Persistent Storage](#)
- [!\[\]\(d4bd0dc972749ad3ba477eac47688a0b_img.jpg\) Apprise API/Web Interface](#)
- [!\[\]\(5eab3de5002abb449199a3fc43c9f414_img.jpg\) Showcase](#)

Want to help make Apprise better?

- [!\[\]\(5ba1bc70d78f05c00988641e5e513c62_img.jpg\) Contribute to the Apprise Code Base](#)
- [!\[\]\(0d3dd579ab24f8020cd6c2659f3acb8c_img.jpg\) Sponsorship and Donations](#)

Revision #2

Created 1 June 2025 14:43:32 by Administrador

Updated 1 June 2025 14:56:29 by Administrador