API Methods: createCheck Endpoint

API Methods: createCheck Endpoint

The CheckCentral API provides an endpoint for programmatically creating Checks.

To create a new Check through the API, an API token with Read/Write access is required. Organization administrators can create tokens through the API portal on your dashboard.

Dashboards ▾   Checks ▾   Notifications ▾   Activities   Organization ▾   Help ▾

API
Ignore Rules
Settings
Stats
Subscription
Users
User Groups

API Tokens + Add Token   🔍 Documentation

no api tokens fou

User API Tokens + Add Token   🔍 Documentation

no api tokens fou

"User API Tokens" will only return data available to the user based on their granted permissions and user groups. Regular "API Tokens" have the Access Level given upon creation.

The endpoint is located at `https://api.checkcentral.cc/createCheck/?apiToken=APITOKEN` where the APITOKEN placeholder is replaced with your valid token.

The request must be made with the `Content-Type` header set to `application/json`. The body of the request should contain as many of the properties below to properly define the Check. Any properties not passed will be set to their default value.

For example, we want to create a new Check that:

- Receives messages at the address `demo+apicheck@mycheckcentral.cc`.

- Expects messages every two hours.

- Expects messages only on weekdays.

- Is set to `Failure` unless the message subject is `Message Success` or the message body has the text `No Errors Reported`.

To achieve this, we send the following json in the body of the request:

```json
{
  "name": "An API Check",
  "email_alias": "apicheck",
  "interval_type": "hour",
  "interval_value": 2,
  "active_days": [
    "mon",
    "tue",
    "wed",
    "thu",
    "fri"
  ],
  "success_conditions": {
    "required_conditions": "any",
    "conditions": [
      {
        "message_field": "subject",
        "matches": "exactly",
        "value": "Message Success"
      },
      {
        "message_field": "body",
        "matches": "contains",
        "value": "No Errors Reported"
      }
    ]
  }
}
```

The check data must include a `name` property, as well as at least one `email_alias` and `matching_conditions` to match the Check against incoming messages.

The complete list of possible properties for check configuration are as follows:

Check Structure

Parameter	Type/Allowed Values	Default
`name` The name for the new check. **This parameter is required.**	String	
`email_alias` The 'plus alias' for the check. This will be used to generate the custom email address for the Check along with your organization account name (account_name+email_alias@mycheckcentral.cc). **At least one** `email_alias` **or** `matching_conditions` **parameter must be specified.**	String	
`matching_conditions` The rules that will be used to match incoming messages. (See the ConditionGroup table below.) **At least one** `email_alias` **or** `matching_conditions` **parameter must be specified.**	ConditionGroup	`{}`
`group_id` The ID of the group to assign this check. (IDs may be retrieved from the getGroups API endpoint.)	String	
`description` A text description for the check.	String	

`interval_type`

The interval unit CheckCentral will use to calculate the expected arrival time of incoming messages.

One of:

- `minute`
- `hour`
- `day`
- `week`
- `month`

`day`

`interval_value`

The number of interval units (see `internal_type` above) used when calculating the expected arrival of incoming messages.

Integer

`1`

`overdue_minutes`

How many minutes after the interval has elapsed CheckCentral will wait before flagging the check as failed.

Integer

`30`

`window_start`

An optional time of day when messages will begin to be received.

String

`00:00`

`window_end`

An optional time of day when messages will cease to be received.

String

`00:00`

`active_days`

Pass an array of days to indicate which days the check should expect to receive messages.

An array containing either `all`, or any of:

- `sun`
- `mon`
- `tue`
- `wed`
- `thu`
- `fri`
- `sat`

`all`

`enabled_notifications`

Pass an array of personal notification services to configure the check to send alerts via those services.

An array containing any of:

- `email`
- `sms`
- `push`
- `pushbullet`
- `pushover`

`[]`

`enabled_integrations`

Pass an array of organization notification services to configure the check to send alerts via those services.

An array containing any of:

- `slack`
- `mattermostwebhook`
- `discordwebhook`
- `googlechatwebhook`
- `slackwebhook`
- `microsoftteamswebhook`
- `customwebhook`
- `jirasoftwarewebhook`
- `zendeskapi`
- `ifttt`
- `webexwebhook`
- `zapierwebhook`
- `rocketchatwebhook`
- `zulipwebhook`
- `asanaapi`
- `externalemails`
- `redmineapi`
- `bugzillaapi`
- `githubapi`
- `gitlabapi`
- `microsofttodolistapi`

`[]`

`notify_returned_to_success`

Checks with this flag will send an alert when they are restored to a success state by an incoming message.

Bool

`true`

`notify_outside_window`

Checks with this flag set to false will only send alerts during the time window and active days configured above.

Bool

`true`

`notify_consecutive_alerts`

This parameter controls when CheckCentral will send alerts about repeated failure or warning messages.

One of:

- `always`
- `first`
- `after_n`
- `every_nth`

`always`

`notify_consecutive_alerts_limit`

If `notify_consecutive_alerts` is set to either `after_n` or `every_nth`, this parameter configures how many consecutive failures or consecutive warnings will trigger alerts.

Integer

`1`

`notify_failure_grace_period`

The number of minutes this check will wait to send alerts about failures. If the check is returned to successful status before this period elapses, any pending alerts will be cancelled.

Integer

`0`

`default_status`

The default status for incoming messages that don't match against any of the conditions below.

One of:

- `success`
- `warning`
- `failure`

`failure`

`success_conditions`

The rules that determine whether incoming messages will be set to success. See the definition of a ConditionGroup below.

ConditionGroup

`{}`

`warning_conditions`

ConditionGroup   `{}`

The rules that determine whether incoming messages will be set to warning.
See the definition of a ConditionGroup below.

`failure_conditions`

ConditionGroup   `{}`

The rules that determine whether incoming messages will be set to failure.
See the definition of a ConditionGroup below.

ConditionGroup

Parameter	Type/Allowed Values	Default
`required_conditions` This parameter controls whether incoming messages must meet any or all of the conditions defined below to be considered a match.	One of: • `all` • `any`	`all`
`condense_whitespace` If this flag is set, all of the conditions defined in this group will treat multiple whitespace characters as a single space.	Bool	`false`
`conditions` The list of conditions for this ConditionGroup to match messages against. See the definition of a Condition below.	An array containing one or more of: [Condition]	`{}`

Condition

Parameter	Type/Allowed Values	Default
`message_field` Which part of the incoming message will be examined for this condition.	One of: • `subject` • `body` • `to` • `cc` • `attachments` • `attachment_text` • `from` • `originally_to`	`subject`
`matches` What matching rule this condition will use.	One of: • `contains` • `not_contains` • `exactly` • `empty` • `not_empty` • `has_attachments` • `has_no_attachments` • `filename_contains` • `filename_not_contains` • `filename_exact` • `text_contains` • `text_not_contains` • `complex_match` • `complex_filename_match` • `complex_text_match`	`contains`
`value` The value to use with the above matching rule and message component.	String	

API Methods: createCheck Endpoint

More API Documentation and Help

A full list of API Methods are available via the API page under your CheckCentral organization.

Log in to your account. Click "Organization" from the main menu, then "API."
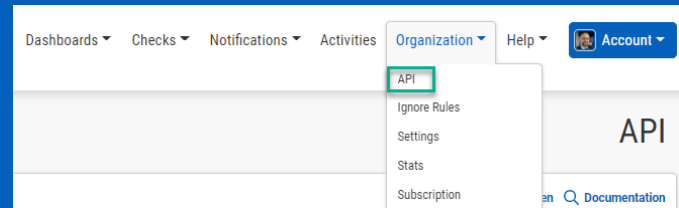
Click "Documentation."
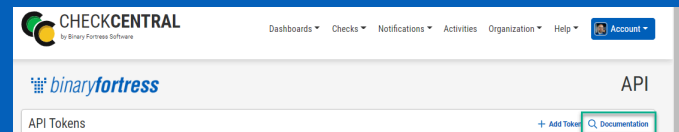
Here you'll see the list of Methods available, their permission requirements, their purpose, and the call limit per day.

Click on any of the Methods to see a testing area with parameter requirements and an example of usage. Populate the fields and click "Send Test Request" to see the JSON Response.

About CheckCentral

CheckCentral Monitoring consolidates and simplifies backup, system, and software email updates into a clean, graphical dashboard, bringing peace of mind to IT administrators of SMBs, Enterprises, and MSPs.

To learn more about CheckCentral, visit: **https://www.checkcentral.cc**

About Binary Fortress Software

Binary Fortress has spent 18 years in pursuit of one goal: create software to make life easier. Our software ranges from display management and system enhancement utilities to monitoring tools and digital signage. IT administrators, professional gamers, coffee-shop owners, and MSPs all rely on Binary Fortress to make their days better, and their lives easier.

Copyright © 2007-2025 Binary Fortress Software, all rights reserved.
The Binary Fortress logo is a trademark of Binary Fortress Software.
The CheckCentral logo is a trademark of Binary Fortress Software.

Binary Fortress Software
1000 Innovation Drive, Suite 500
Kanata, Ontario, Canada
K2K3E7
https://www.binaryfortress.com