Using Apps Script to Build a Google Chatbot to Schedule Messages

Michelle Melton
Appalachian State University

@unccause #unccause2022

Introduction

About Me

Michelle Melton

Lead Programmer-Developer

Academic Technologies, ITS

Appalachian State University

meltonml@appstate.edu

@haietza

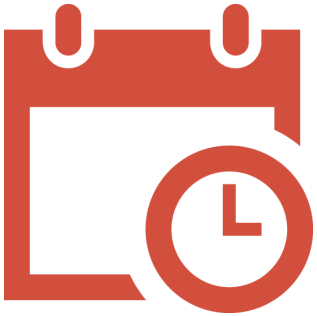
47.8%

use group chat tools to communicate with coworkers

The Use of Group Chat Tools at Work, SpiderOak

While the ability to communicate and collaborate more quickly is certainly an advantage in today's workplace, there are times when sending an instant message may not be ideal.

Use Cases



Planning ahead



Avoiding distractions



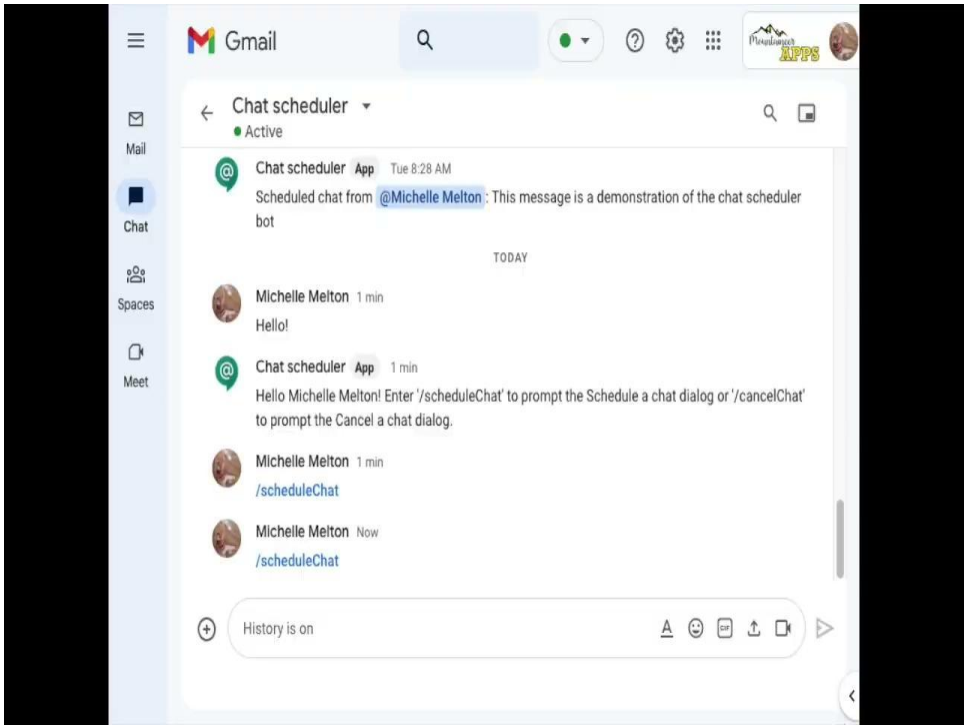
Following up

The Chat Scheduler chatbot allows users to schedule a message to a chat or space for a specified time in the future.

Overview

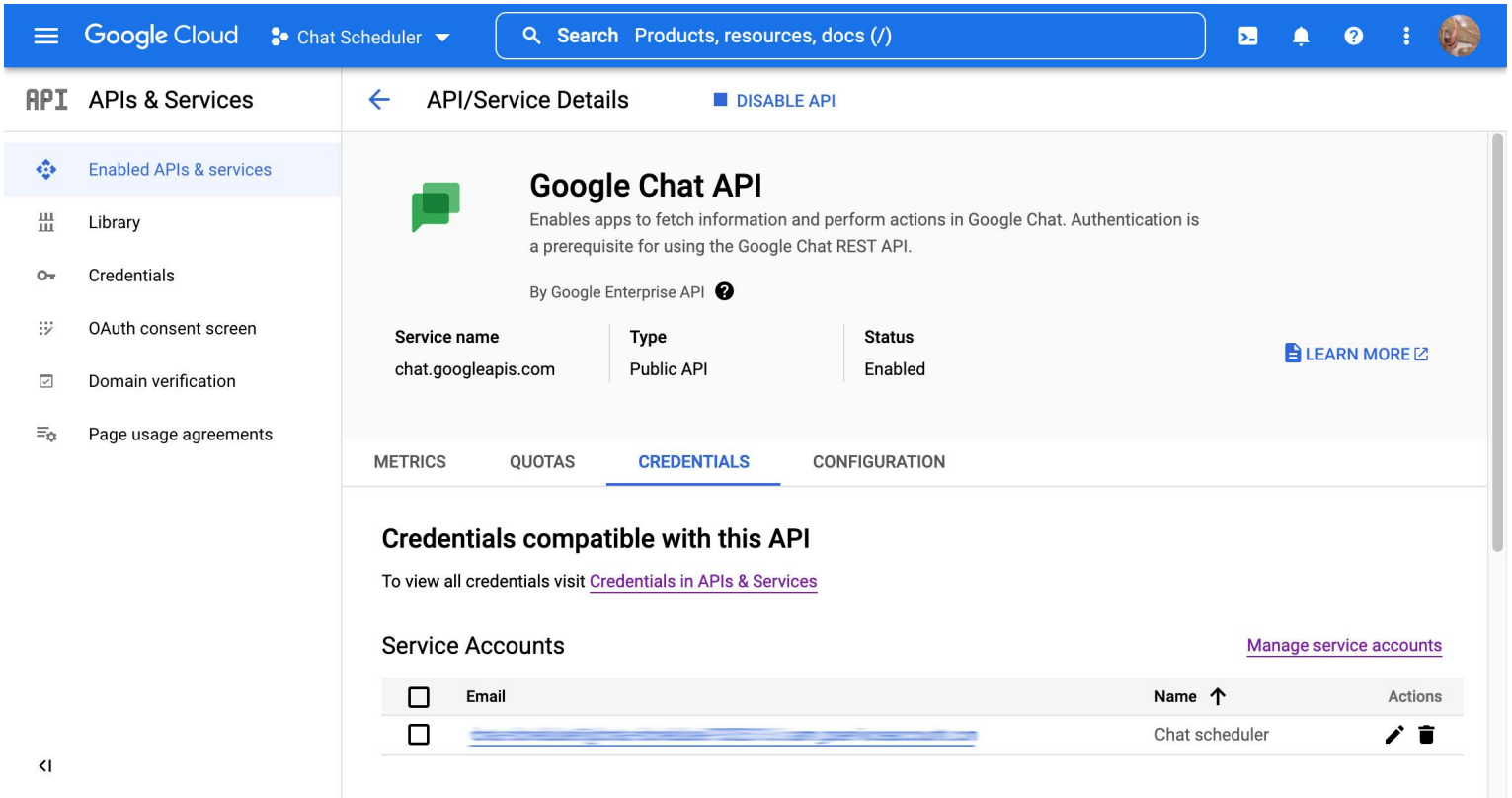
- Demonstration
- Google Cloud Project
- Apps Script Project

Demonstration

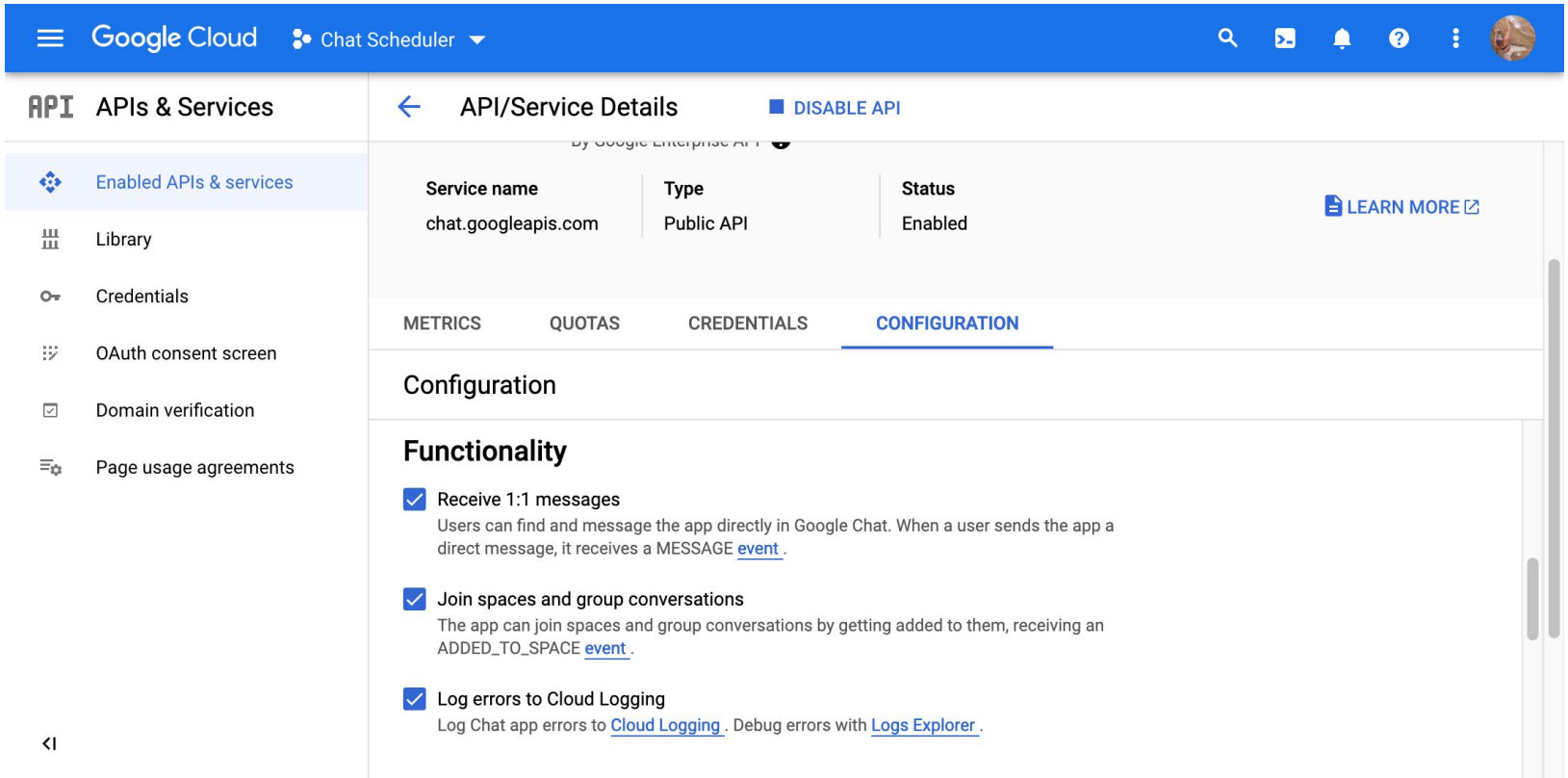


Google Cloud Project

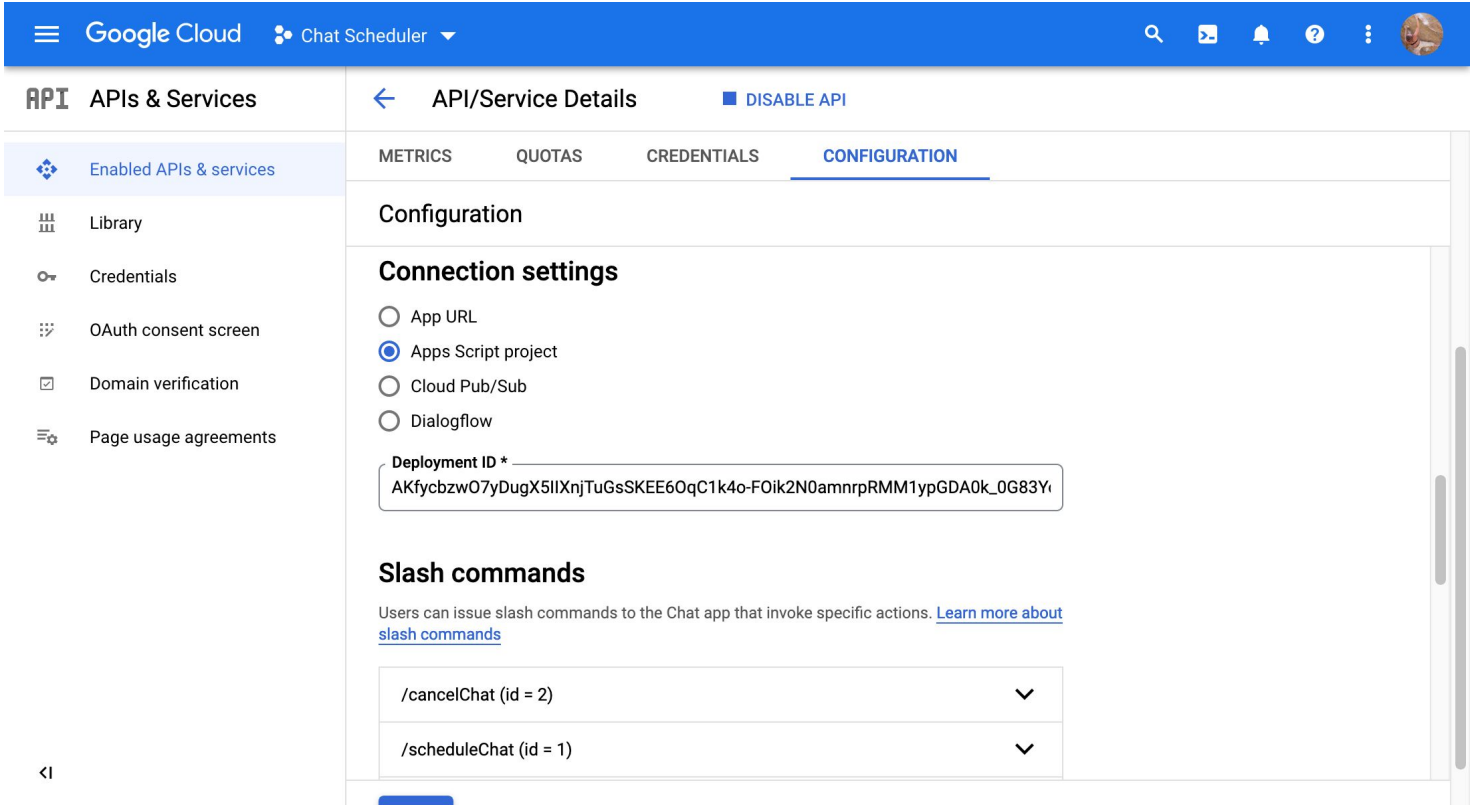
Google Cloud Project



Google Cloud Project

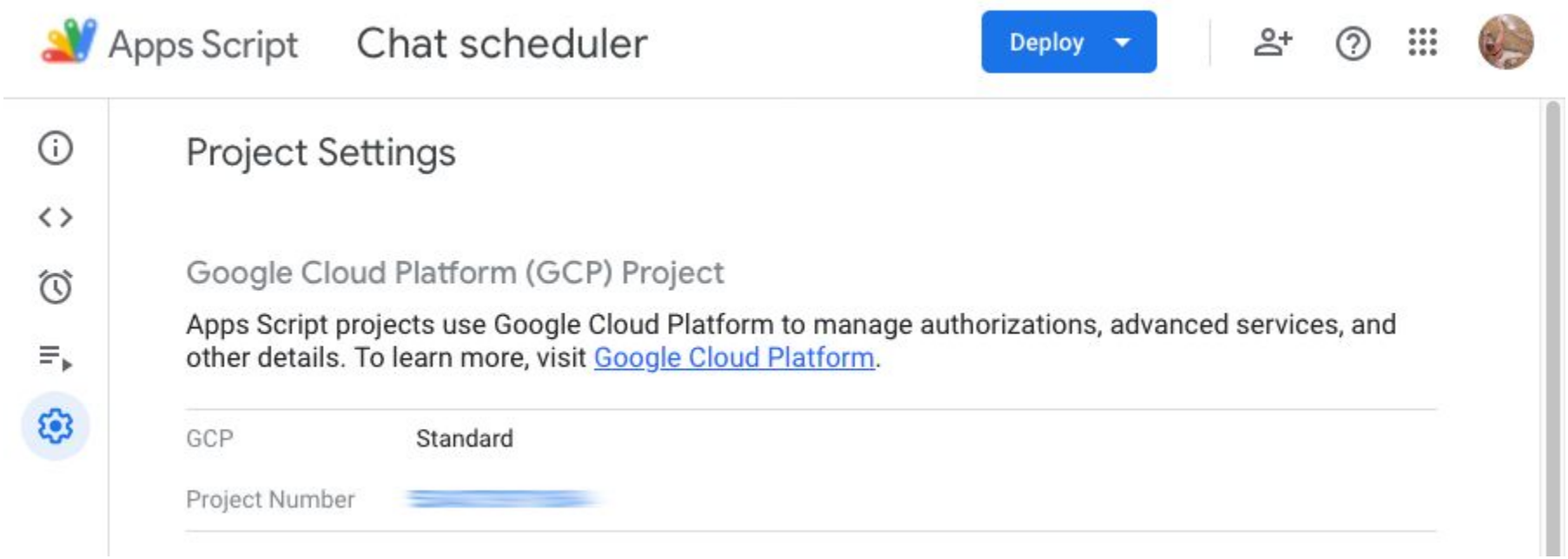


Google Cloud Project

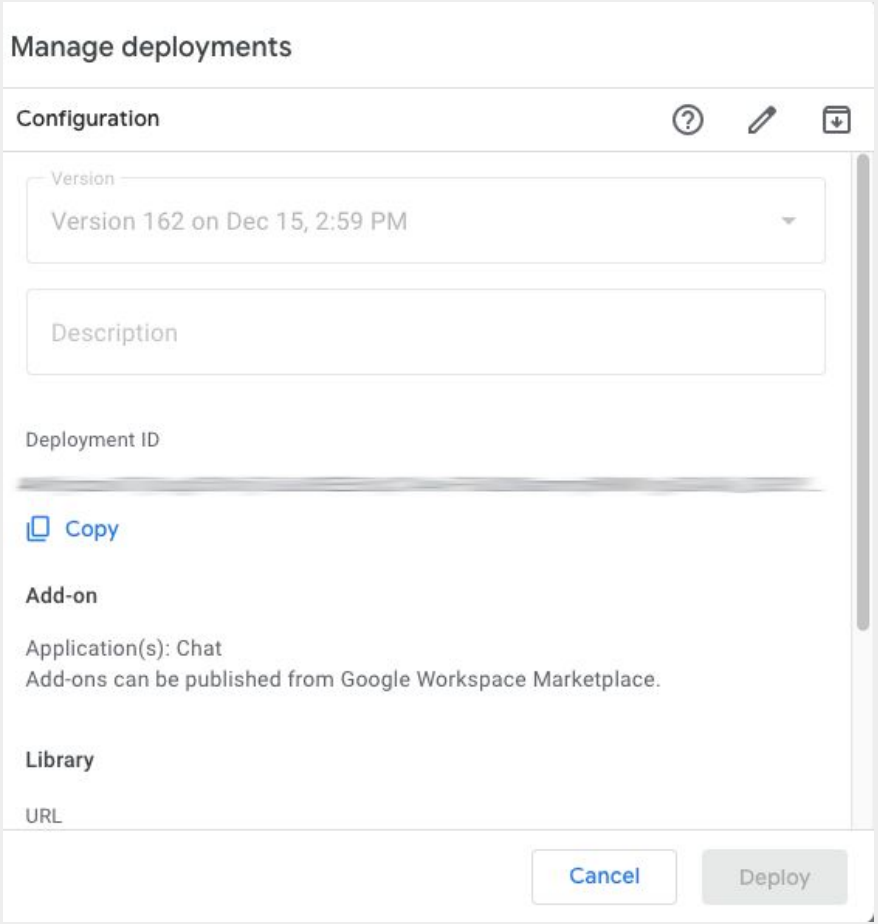


Apps Script Project

Project Settings



Deployment


Manage deployments
Configuration
Version
Version 162 on Dec 15, 2:59 PM
Description
Deployment ID
Copy
Add-on
Application(s): Chat
Add-ons can be published from Google Workspace Marketplace.
Library
URL
Cancel
Deploy

Start a Conversation

- Start a conversation with the bot
- Add the bot to a conversation with other users
- Add the bot to a space

Start a Conversation

```
function onAddToSpace(event) {
 let message = "";

 if (event.space.singleUserBotDm) {
   message = "Thank you for adding me to a DM, "
     + event.user.displayName + "!";
 } else {
   message = "Thank you for adding me to " +
       (event.space.displayName ? event.space.displayName : "this chat");
 }
```

Start a Conversation

```
let properties = PropertiesService.getScriptProperties().getProperties();
let alreadyIncluded = false;
for (let property in properties) {
  if (property == event.space.name) {
    alreadyIncluded = true;
    break;
  }
}
```

Start a Conversation

```
// Store space in script properties as available recipient.
if (!alreadyIncluded) {
  let memberNames = listMembers(event.space.name);
  let conversationName = memberNames.join(", ");
  let displayName = event.space.displayName ? event.space.displayName :
conversationName;
  PropertiesService.getScriptProperties().setProperty(event.space.name,
    displayName);
}
```

Schedule a Chat

```
function onMessage(event) {
 ...
   return { "text": "Hello " + event.user.displayName +
     "! Enter '/scheduleChat' to prompt the Schedule a chat dialog " +
     "or '/cancelChat' to prompt the Cancel a chat dialog." }
 }
}
```

Schedule a Chat

```
function onMessage(event) {
 if (event.message.slashCommand) {
   switch (event.message.slashCommand.commandId) {
     case 1: // Command Id 1, scheduleChat
       return getScheduleCard(event.user.name, event.space.name);
...
```

Schedule a Chat

```
function getScheduleCard(userName, spaceName) {
 let properties = PropertiesService.getScriptProperties().getProperties();
 ...
 for (let property in properties) {
   if (property.includes("spaces/") && isMember(property, userName)) {
     if (property == spaceName) {
       option = {"text": properties[property], "value": property, "selected": true};
     } else {
       option = {"text": properties[property], "value": property, "selected": false};
     }
     spaceOptions.push(option);
   }
 }
```

Schedule a Chat

```
return {
   "action_response": {
     "type": "DIALOG",
     "dialog_action": {
       "dialog": {
         "body": {
           "sections": [
             {
               "widgets": [
...
```

Schedule a Chat

```
{
  “selectionInput": {
    "type": "DROPDOWN",
    "label": "Select the chat or space recipient",
    "name": "recipient",
    "items": spaceOptions
    },
},
```

Schedule a Chat

```
{
  "textInput": {
    "label": "Message",
    "type": "MULTIPLE_LINE",
    "name": "message",
    "hintText": "Enter the message to be sent",
    "value": ""
  }
},
```

Schedule a Chat

```
{
  "selectionInput": {
    "type": "DROPDOWN",
    "label": "Month",
    "name": "month",
    "items": [
      buildMonth("January", "0"),
      buildMonth("February", "1"),
      ...
    ]
  },
},
```

Schedule a Chat

```
function buildMonth(monthText, monthNumber) {
 const now = new Date();
 let month = now.getMonth();
 if (month == monthNumber) {
   return {"text": monthText, "value": monthNumber, "selected": true};
 } else {
   return {"text": monthText, "value": monthNumber, "selected": false};
 }
}
```

Schedule a Chat

```
"name": "Schedule a chat",
"fixedFooter": {
  "primaryButton": {
    "text": "Schedule chat",
    "onClick": {
      "action": {
        "function": "scheduleChat"
      }
    },
    "altText": "Schedule chat",
  }
}
```

Schedule a Chat

```
function onCardClick(event) {
 ...
 if (event.action.actionMethodName == 'scheduleChat') {
   return scheduleChat(event);
 }
 ...
```

Schedule a Chat

```
function scheduleChat(event) {
  let now = new Date();
  let year = now.getFullYear();
  let month = parseInt(event.common.formInputs.month[''].stringInputs.value[0]);
  ...
  let request = {"text": "Scheduled chat from <" + event.user.name + ">: " + message}
  let messageObj = {"sender": event.user.name, "recipient": recipient, "
    time": time.getTime(), "request": request, "message": message}

 // Add message object to script properties for asynchronous scheduling.
 PropertiesService.getScriptProperties()
     .setProperty(time.getTime(), JSON.stringify(messageObj));
```

Cancel a Chat

```
function onMessage(event) {
 ...
   return { "text": "Hello " + event.user.displayName +
     "! Enter '/scheduleChat' to prompt the Schedule a chat dialog " +
     "or '/cancelChat' to prompt the Cancel a chat dialog." }
 }
}
```

Cancel a Chat

```
function onMessage(event) {
 if (event.message.slashCommand) {
   switch (event.message.slashCommand.commandId) {
     ...
     case 2: // Command Id 2, cancelChat
       return getCancelCard(event.user.name);
```

Cancel a Chat

```
function getCancelCard(userName) {
 let properties = PropertiesService.getScriptProperties().getProperties();
 let messages = [];
 for (let property in properties) {
   if (!isNaN(property)) {
     let messageObj = JSON.parse(properties[property]);
     if (messageObj.sender == userName) {
       let message = {"text": messageObj.message, "value": property,
         "selected": false};
       messages.push(message);
     }
   }
 }
```

Cancel a Chat

```
{
  "selectionInput": {
    "type": "DROPDOWN",
    "label": "Select the message to cancel",
    "name": "messages",
    "items": messages
  }
}
```

Cancel a Chat

```
"name": "Cancel a chat",
"fixedFooter": {
  "primaryButton": {
    "text": "Cancel chat",
    "onClick": {
      "action": {
        "function": "cancelChat"
      }
    },
    "altText": "Cancel chat",
  }
}
```

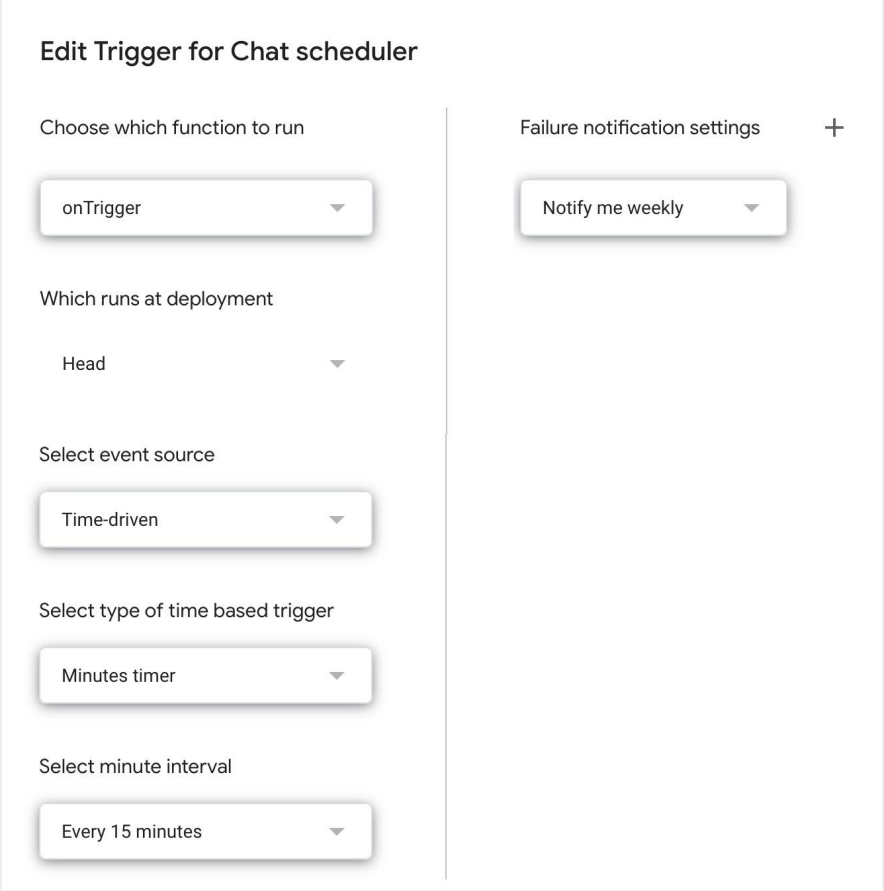
Cancel a Chat

```
function onCardClick(event) {
 ...
 } else if (event.action.actionMethodName == 'cancelChat') {
   return cancelChat(event);
 }
}
```

Cancel a Chat

```
function cancelChat(event) {
 let message = event.common.formInputs.messages[''].stringInputs.value[0];
 PropertiesService.getScriptProperties().deleteProperty(message);
 return {
   "action_response": {
     "type": "DIALOG",
     "dialog_action": {
       "action_status": "OK"
     }
   }
 };
}
```

Triggers


Edit Trigger for Chat scheduler
Choose which function to run
onTrigger
Which runs at deployment
Head
Select event source
Time-driven
Select type of time based trigger
Minutes timer
Select minute interval
Every 15 minutes
Failure notification settings
Notify me weekly

Triggers

```
function onTrigger() {
 let today = new Date();
 let now = today.getTime();
 let properties = PropertiesService.getScriptProperties().getProperties();
 for (let property in properties) {
   // Messages are stored in script properties with time as index
   if (!isNaN(property)) {
     let message = JSON.parse(properties[property]);
     if (message.time < now) {
       postMessage(message.recipient, message.request, property);
     }
   }
 }
}
```

Triggers

```
function postMessage(spaceId, request, time) {
 let service = OAuth2.createService('chat')
     .setTokenUrl('https://accounts.google.com/o/oauth2/token')
     .setPrivateKey(SERVICE_ACCOUNT_PRIVATE_KEY)
     .setClientId(SERVICE_ACCOUNT_EMAIL)
     .setPropertyStore(PropertiesService.getUserProperties())
     .setScope(SCOPE);
 if (!service.hasAccess()) {
   Logger.log('Authentication error: %s', service.getLastError());
   return;
 }
```

Triggers

```
let url = 'https://chat.googleapis.com/v1/' + spaceId + '/messages';
 UrlFetchApp.fetch(url, {
   method: 'post',
   headers: { 'Authorization': 'Bearer ' + service.getAccessToken() },
   contentType: 'application/json',
   payload: JSON.stringify(request),
 });

 PropertiesService.getScriptProperties().deleteProperty(time);
}
```

Remove Bot from Conversation

```
function onRemoveFromSpace(event) {
 console.info("Bot removed from ",
     (event.space.name ? event.space.name : "this chat"));

 // Remove space from script properties, no longer available recipient.
 PropertiesService.getScriptProperties()
     .deleteProperty(event.space.name);
}
```

Review

- Demonstration
- Google Cloud Project
- Apps Script Project

The Chat Scheduler chatbot provides a solution for when sending an instant message is not ideal.

Discussion