

Auto Add all Localizations to a Release

This guide will show you how to set up an automation that automatically adds the entry and all locales to a target release. Instead of a user going through each language and adding them one by one, this will add each current version of each locale to the release. Follow the step-by-step instructions and code snippets to get started.

Block 1: Workflow Trigger

Description: This automation triggers whenever you have a workflow stage change to a certain state. This trigger can be adjusted to meet your organization's criteria.

Step-by-Step Configuration

Add the Block: HTTP - HTTP Request

Configure Variables:

- Stack
 - Description: Which stack you want this to work off of.
- Branch
 - Description: Default is `main` but change accordingly.
- Content Type
 - Description: The content type you want this to work for. If left blank, it will work for all content types.
- Workflow
 - Description: The workflow that you will be using for this.
- Workflow Stage
 - Description: The workflow stage you want this to trigger on, for example, a stage named "localize".

Block 2: HTTP Request

Description: Fetch all the available locales for this entry when you localize it using an HTTP block that interacts with the Content Management API.

Step-by-Step Configuration

Add the Block: HTTP - HTTP Request

Configure Variables:

- URL
 - Description: The API Endpoint to retrieve locales. This query contains dynamic variables from previous steps, so adjust these values as needed.

JavaScript

```
https://api.contentstack.io/v3/content_types/{{1.body.data.workflow.content_type.oid}}/entries/{{1.body.data.workflow.entry.oid}}/locales
```

- Method
 - Description: Set the value here to **GET**

Block 3: Get a Single Release

This is currently hardcoded. You will need to find a way to configure with release you want to use.

Step-by-Step Configuration

1. **Add the Block:** Contentstack - Get a Single Release
2. **Configure Variables:**
 - Stack
 - Description: The API Endpoint to retrieve locales. You can see that there are dynamic variables in this query that pull from previous steps, so you may need to change these in the platform to your exact value
 - Branch
 - Description: The current branch you are working, default is Main.
 - Release
 - The release you want to use. This is hardcoded, you will need to find a way to change this dynamically.

Block 4: Code Block

This is where a majority of the logic comes into play, on iterating an entry for locales that are localized and adding them to the hardcoded release.

Step-by-Step Configuration











3. Add the Block: Codeblock - Javascript Code

4. Configure Variables:

- Input
 - Description: These are all the inputs you will be needed for the code block to work. They take values from previous steps that we will use in the javascript snippet to function properly. You will need to add your management token manually to the value as well.

Input

Enter the input name and value you want to use in your JavaScript code. You can also access the data from previous steps.

Input Name	Input Value	
entryuid	 1.body.data.workflow.entry.uid x	
contenttypeuid	 1.body.data.workflow.content_type.uid x	
releaseuid	 3.release.uid x	
apikey	 1.body.api_key x	
managementtoken		
<div>+ Add Input</div>		

JavaScript

```
let log = [];  
const fetchLocales = async () => {
```

```

    try {
      let url =
`https://api.contentstack.io/v3/content_types/${input.contenttypeid}/entries/${
input.entryuid}/locales`;
      const response = await fetch(

`https://api.contentstack.io/v3/content_types/${input.contenttypeid}/entries/${
input.entryuid}/locales`,
      {
        method: "GET",
        headers: {
          authorization: input.managementtoken,
          api_key: input.apikey, // Assuming this is the required header for
auth
          "Content-Type": "application/json",
        },
      },
    );

    if (!response.ok) {
      throw new Error(`Error fetching locales: ${response.statusText}`);
    }

    const data = await response.json();
    return data.locales; // Assuming the response JSON has a 'locales' field
containing the array of locales
  } catch (error) {
    console.error("Error:", error);
  }
};

const fetchLocaleEntry = async (locale) => {
  try {
    const response = await fetch(

`https://api.contentstack.io/v3/content_types/${input.contenttypeid}/entries/${
input.entryuid}?locale=${locale.code}`,
    {
      method: "GET",
      headers: {
        authorization: input.managementtoken,
        api_key: input.apikey,
        "Content-Type": "application/json",
      },
    },
  ),

```

```

    },
  );

  if (!response.ok) {
    throw new Error(
      `Error fetching entry for locale ${locale.code}:
      ${response.statusText}`,
    );
  }

  const data = await response.json();
  return data.entry;
} catch (error) {
  console.error("Error:", error);
}
};

const sendLocaleToRelease = async (locale) => {
  if (locale.localized || locale.code === "en-us") {
    const entryData = await fetchLocaleEntry(locale);
    console.log(
      JSON.stringify({
        item: {
          version: entryData._version,
          uid: input.entryuid,
          content_type_uid: input.contenttypeuid,
          action: "publish",
          locale: locale.code,
        },
      }),
    );
    try {
      const response = await fetch(
        `https://api.contentstack.io/v3/releases/${input.releaseuid}/item`,
        {
          method: "POST",
          headers: {
            authorization: input.managementtoken,
            api_key: input.apikey,
            "Content-Type": "application/json",
          },
          body: JSON.stringify({
            item: {
              version: entryData._version,

```

```

        uid: input.entryuid,
        content_type_uid: input.contenttypeuid,
        action: "publish",
        locale: locale.code,
    },
   )),
},
);

if (!response.ok) {
    throw new Error(
        `Error sending locale to release: ${response.statusText}`,
    );
}
const data = await response.json();
console.log("Success: ", data);
} catch (error) {
    console.error("Error:", error);
}
}
};

const processLocales = async () => {
    const locales = await fetchLocales();
    if (locales && Array.isArray(locales)) {
        for (const locale of locales) {
            await sendLocaleToRelease(locale);
        }
    } else {
        console.error("No locales found or invalid data format");
    }
    return JSON.stringify(log);
};

// Start the process
return processLocales()

```

