2019年12月24

教師画像データの効率的な蓄積のプログラム詳細設計書

（部 署）　IoT Solutions Center Production Solutions Group

（氏 名）陳昭成

改定履歴

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rev. | 日付 | 承認者 | 審査者 | 作成者 | 内　容 |
| 1 | 2019/11/20 | 花田 | 鈴木 | 陳 | 初版として作成 |
| 2 | 2019/11/25 | 花田 | 鈴木 | 陳 | システム設計を完成 |
| 3 | 2019/11/26 | 花田 | 鈴木 | 陳 | システム設計を修正 |
| 4 | 2019/11/28 | 花田 | 鈴木 | 陳 | プログラムで使っている  Class, Functionsの説明を追加 |
| 5 | 2019/12/09 | 花田 | 鈴木 | 陳 | 二つの機能を追加   1. 削除する機能 2. 圧縮済みフォルダー記録 |
| 6 | 2019/12/11 | 花田 | 鈴木 | 陳 | 追加された機能のタイトルを修正 |
| ７ | 2019/12/17 | 花田 | 鈴木 | 陳 | 圧縮されたか確認するメカニズムの詳細設計説明を追加 |
| 8 | 2019/12/20 | 花田 | 鈴木 | 陳 | 削除する機能を修理 |
| 9 | 2019/12/24 | 花田 | 鈴木 | 陳 | 圧縮済みフォルダー記録のバックアップファイルはXMLからJSONというフォーマットに変更 |

1. プログラムタイプ：C# Windows Service

Windowsサービスとは、起動時に自動で開始され、Windows PCの稼働中にバックグラウンドで働き続けるプログラムのことを指します。（参考：http://minto.tech/windows-service/）

＊ここから、教師画像データの効率的な蓄積のプログラムはAutoCompressorWindowsServiceと呼びます。

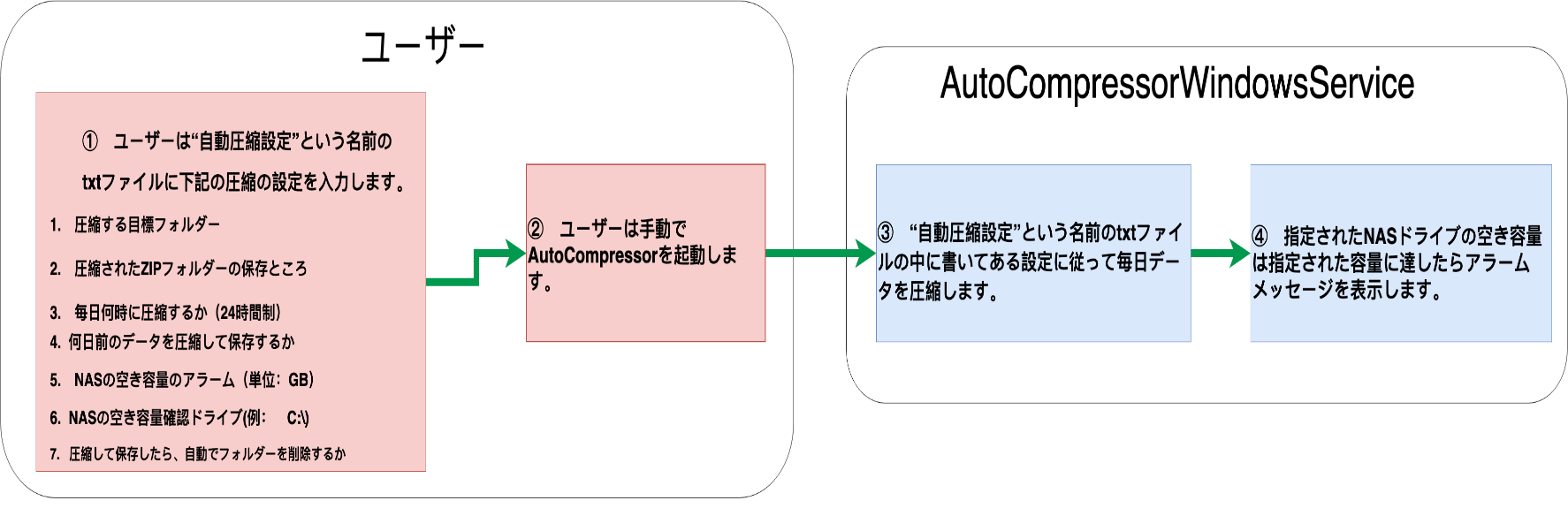
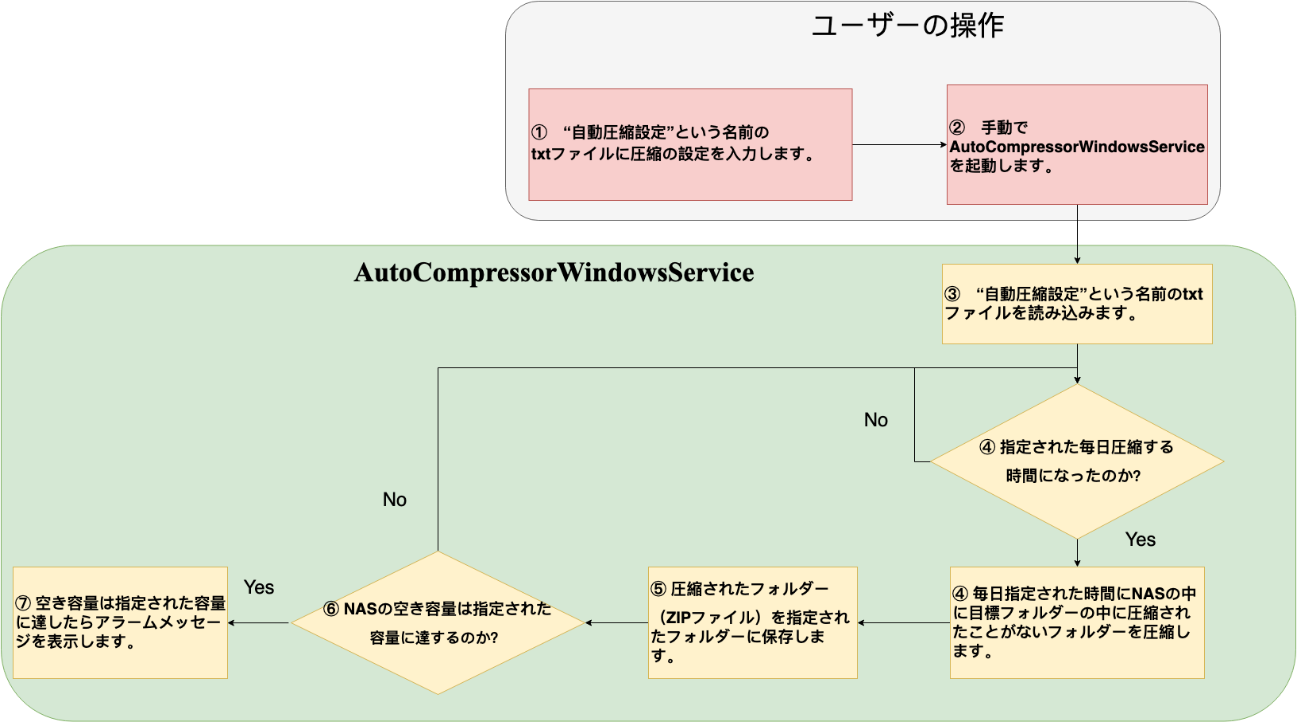
実行フローチャート

図 1．実行フローチャート

1. システムフローチャート：

We will call定期的に自動で圧縮して効率的に保存できるプログラム“AutoCompressorWindowsService” in the following part.



The followings are the design of each steps in the flowchart above.

The design of ①：

Step 1 We create a txt file and name it “自動圧縮設定.txt”

Step 2 Allow the user to edit the settings of the AutoCompressorWindowsService

The user is allowed to set and save the following settings of the AutoCompressorWindowsService in the “自動圧縮設定.txt”.

1.　圧縮する目標フォルダー(target folder)

2.　圧縮されたZIPフォルダーの保存先(ZIP storage folder)

3.　毎日何時に圧縮するか（24時間制）(compression time)

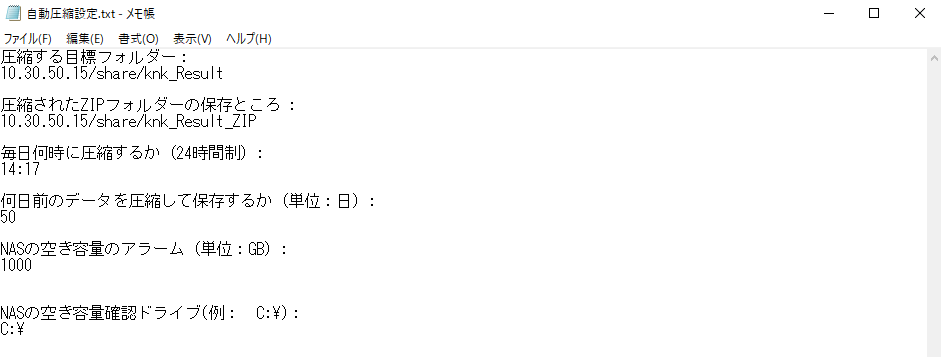
4. 何日前のデータを圧縮して保存するか（単位：日）(compress the folder created over XX days ago)

5.　NASの空き容量のアラーム（単位：GB）(free space limit)

6.　NASの空き容量確認ドライブ(例：　C:\)

7．圧縮して保存したら、自動でフォルダーを削除するか (Yes or No を入力してください)

For example, the user can edit the settings like the following.



In this case, the AutoCompressorWindowsService will compress the folders in the “10.30.50.15/share/knk\_Result\_ZIP” folder that have not been compressed and are created over 50 days ago at 14：17 every day.

The compressed folders (ZIP files) will be stored in the

“10.30.50.15/share/ knk\_Result” folder. In addition, when the remaining free NAS space of drive C reaches 1000GB, the AutoCompressorWindowsService will display a warning message to remind the engineer.

The design of ②：

Step 1 Every time after the user modify the “自動圧縮設定”.txt, the user needs to manually start the AutoCompressorWindowsService, if the AutoCompressorWindowsService is currently running, the user needs to manually restart it.

Step 2 The AutoCompressorWindowsService will be run in the background.

The design of ③④⑤：

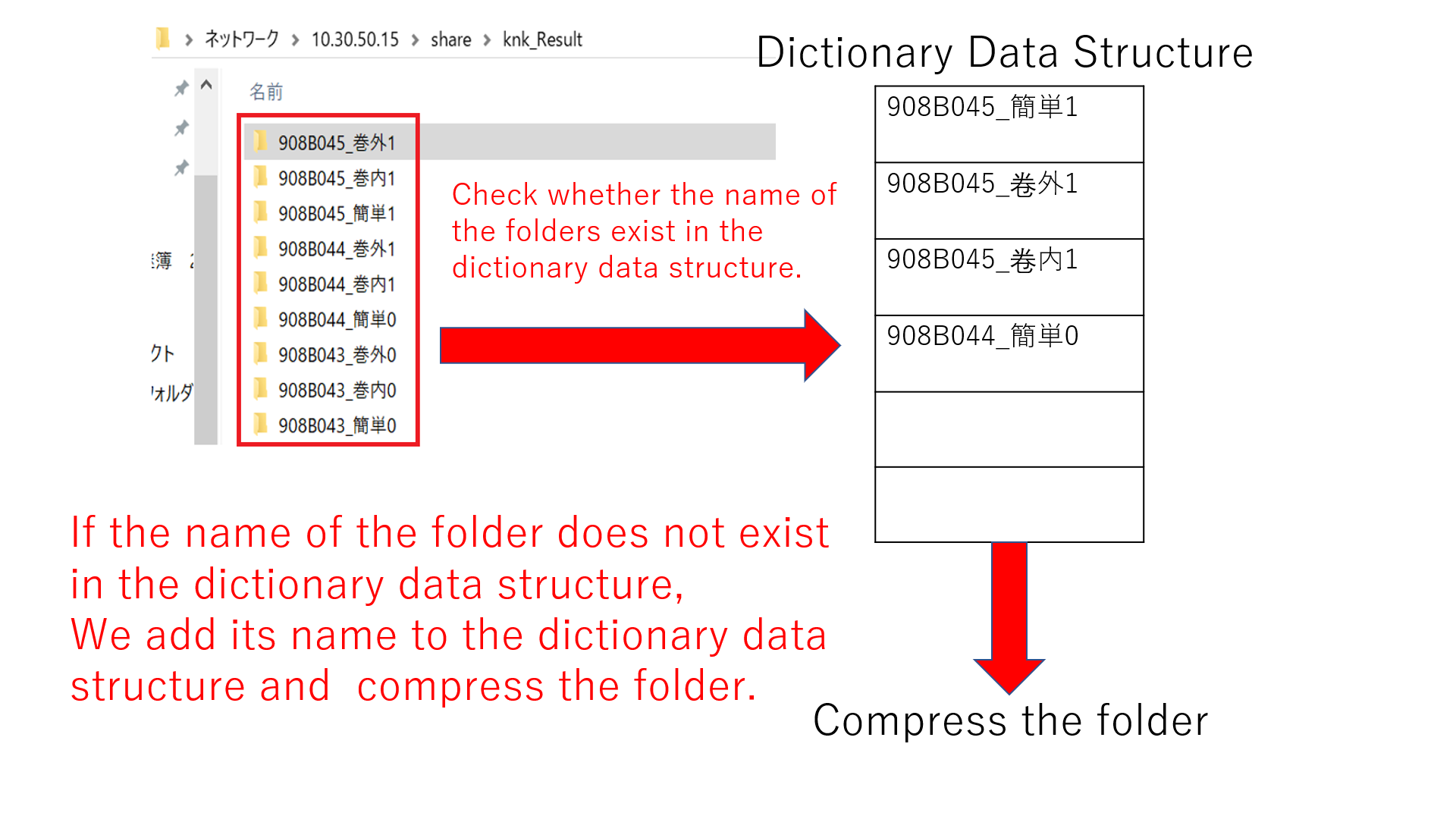
Step 1 The AutoCompressorWindowsService will read in the settings in the “自動圧縮設定”.txt.

Step 2 The AutoCompressorWindowsService sets a timer according to the user’s settings

Step 3 When the time comes, the AutoCompressorWindowsService compresses the folders that have not been compressed in the target folder(圧縮する目標フォルダー) set by the user in the“自動圧縮設定”.txt.

Step 3-1 We use a dictionary data structure in the AutoCompressorWindowsService to save the name of the folders that have been compressed.

Step 3-2 We input all the name of the folders in the target folder (圧縮する目標フォルダー) to the dictionary data structure, if the name doesn’t exist in the dictionary data structure, we compress the folder of that name.



Step 4 We store the compressed files(ZIP files) to the ZIP storage folder (圧縮されたZIPフォルダーの保存ところ)set by the user in the“自動圧縮設定”.txt

The design of ⑥⑦：

Step 1 The AutoCompressorWindowsService checks free NAS space

Step 2 If the remaining free NAS space reaches the free space limit(NASの空き容量のアラーム) set by the user in the“自動圧縮設定.txt”, the AutoCompressorWindowsService displays an alarm message.

1. Program Steps Design:

The user is allowed to set and save the following settings of the AutoCompressorWindowsService in the “自動圧縮設定.txt”.

1.　圧縮する目標フォルダー(target folder)

2.　圧縮されたZIPフォルダーの保存先(ZIP storage folder)

3.　毎日何時に圧縮するか（24時間制）(compression time)

4. 何日前のデータを圧縮して保存するか（単位：日）(compress the folder created over XX days ago)

5.　NASの空き容量のアラーム（単位：GB）(free space limit)

6.　NASの空き容量確認ドライブ(例：　C:\)

7．圧縮して保存したら、自動でフォルダーを削除するか (Yes or No を入力してください)

Step 1 Read in the “自動圧縮設定.txt”

Step 1-1 Record all the request set by the user in the “自動圧縮設定.txt”

Step 2 Activate the compression automatically according to the compression time set by the user(毎日何時に圧縮するか（24時間制）)

When the compression time comes.

Step 3 Get folders created over XX days ago (何日前のデータを圧縮して保存するか)　in the target folder(圧縮する目標フォルダー)

according to the user’s request.

Step 3-1 Get creation time of each folder

Step 3-2 Get the folders that created over XX days ago

Step 4 Compress the folders created over XX days ago (何日前のデータを圧縮して保存するか)　in the target folder(圧縮する目標フォルダー)

Step 5 Save the compressed folder to the ZIP storage folder（圧縮されたZIPフォルダーの保存先）according to the user’s request.

Step 6 Delete the folders that have been compressed to zip files and saved

Step 7 Check the remaining free NAS space

Step 7-1 If the remaining free NAS space reaches the free space limit(NASの空き容量のアラーム) , display the warning message to notify the user.

1. Software Design:

Class Design:



1. Class specification and propose:

Main.cs Class member functions introduction:

Purpose:

1. Activate functions in the ReadInUserSettings.cs to read in the user’s settings.

2. When time comes, activate functions in the AutoCompressor.cs to compress,check free disk space,display warning message.

Class member functions:

OnStart(Windows Service Default function):

This function will be activated when the AutoCompressorWindowsService is activated by the user.

Set up a timer that ticks every 1 minute.

OnStop(Windows Service Default function):

When the AutoCompressorWindowsService is deactivated,

Stop and reset the timer

AutoFolderCompressorTimer\_Elapsed:

Every time the timer ticks,

Check whether the compression time set by the user comes

If yes,

｛

Step １．Compress folders according to the user's settings

Step ２．Check the free disk space

｛

IsCompressionTime：

Check whether the compression time set by the user comes.

compressFolderAccordingToSettings:

Compress folders according to the user's settings.

checkFreeDiskSpace:

Check the free disk space

showMsgBoxFromWS:

Display GUI message from Windows Service

resetFolderStatusAfterCompressLog:

Clear the content of the folderStatusAfterCompressLog

AutoCompressor.cs Class member functions introduction:

Purpose:

1.Compress folders according to the user’s settings.

2.Check NAS free disk space

Class member functions:

compressFolder:

Compress the folders created over XX days ago (何日前のデータを圧縮して保存するか)

and have not been compressed yet in the target folder(圧縮する目標フォルダー).

createZIPFile:

Compress the folder sent from function “compressFolder” to create a ZIP file.

getDiskFreeSpace:

Get free space of the drive selected by the user.

ReadInUserSettings.cs Class member functions introduction:

Purpose:

Read in the 自動圧縮設定.txt.

Class member functions:

ReadInUserSettings(It is the constructor of the class ” ReadInUserSettings”):

Assign the values read from the 自動圧縮設定.txt for the settings of the AutoCompressorWindowsService.

readInUserSettingsFromTxt:

Read in the user's settings from the 自動圧縮設定.txt.

getTargetFolder:

Return the value of 圧縮する目標フォルダー

getZIPStorageFolder:

Return the value of 圧縮されたZIPフォルダーの保存ところ

getCompressionTime:

Return the value of 毎日何時に圧縮するか（24時間制）

getFolderOverNDays:

Return the value of 何日前のデータを圧縮して保存するか

getFreeSpaceLimit:

Return the value of NASの空き容量のアラーム（単位：GB）

getNASDriveName:

Return the value of NASの空き容量確認ドライブ(例：　C:\)

CheckRemoteDriveFreeSpace.cs Class member variables introduction:

Purpose:

Check the free space of a drive on a remote PC or NAS.

Class member variables:

long getRemoteDriveFreeSpace(string folderName):

Return the free space of a remote drive in byte

DynamicConstants.cs Class member variables introduction:

Purpose:

Store all the variables that might need to be changed when developer deploys the software to a computer.

When we deploy the AutoCompressorWindowsService to another computer we might need to change the variables in the DynamicConstants.cs.

Class member variables:

String userAutoCompressorSettingsTxtFile:

to record the txt file that contains the user's settings of the AutoCompressorWindowsService.

String checkCompressionTimeInterval(Currently, it is set to be 1 minute):

to record the time interval for checking whether compression time comes.

2019/11/29に追加された2つの機能:

１．圧縮されて保存できた教師画像データを削除する機能

AutoCompressorWindowsServiceが教師画像データを圧縮して保存できたことを確認してから、元の教師画像データを削除します。

今はデータを圧縮して保存できたことを確認できるメカニズムを考えています。

２．Output dictionary to txt and read in dictionary content from txt when the user reboots the AutoCompressorWindowsService

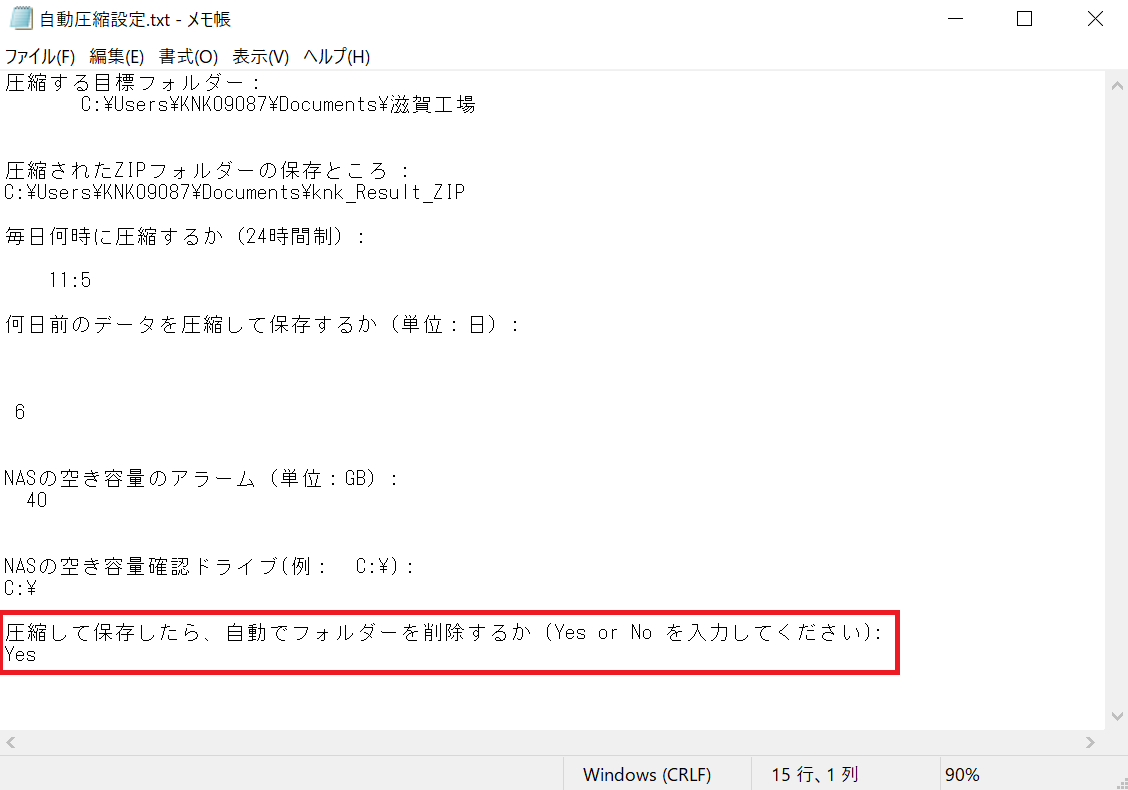
機能1について：

機能１の目的：

フォルダーを圧縮してZIPとして保存してから、元のフォルダー自動で削除する。

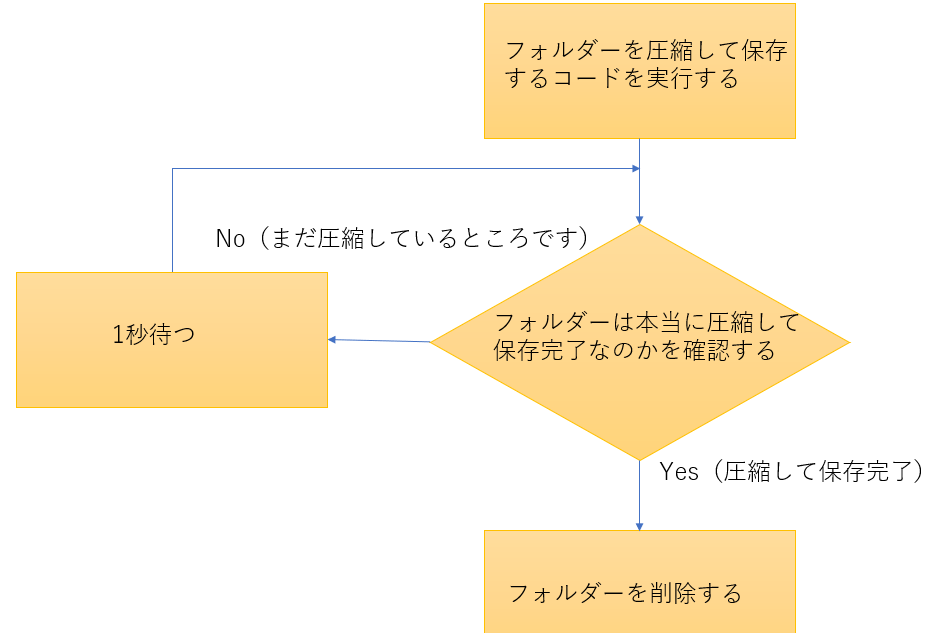
解決策：

今AutoCompressorWindowsService（自動圧縮プログラム）はユーザーが“自動圧縮設定.txt”に設定した” 圧縮して保存したら、自動でフォルダーを削除するか (Yes or No を入力してください): “のYes, Noによって、フォルダーを圧縮したら、元のフォルダーを削除するかを決めます。



もし、ユーザーは“自動圧縮設定.txt”に設定した” 圧縮して保存したら、自動でフォルダーを削除するか (Yes or No を入力してください): “の設定はYesだったら、

下記のフローチャートの通り、元のフォルダーを削除する時に、プログラムはそのフォルダーの中のファイルはまだ圧縮されているかどうかを確認します。もし、まだ圧縮されている場合は1秒待ってから、もう一度ファイルはまだ圧縮されているかどうかを確認します。このようにフォルダーの内容が完全に圧縮されたまでに繰り返して待ちます。完全に圧縮されてから、フォルダーを削除します。



**圧縮されたか確認する(機能1)メカニズムの詳細設計**：

The AutoCompressorWindowsService will try to open each file in the folder with “FileShare=None” setting.

It means no user or program can access this file at this moment.

(下記のAutoCompressorWindowsServiceのsource codeのscreenshotの赤色のところ)

例えば、添付AutoCompressorWindowsServiceのsource codeをwordで開いている日本語コミュニケーション強化研修.docというwordファイルに適用したら、

“C:\Users\KNK09087\Documents\deletetest\日本語コミュニケーション強化研修.doc is being used now.”というメッセージが出てきます。（このメッセージ表示は分かりやすく説明するために追加したのです。AutoCompressorWindowsServiceにはありません。）

If the file open fails, it means the file is being used. In our case it means the file is still being compressed.

The AutoCompressorWindowsService will wait for 1 second and try to open the file again.

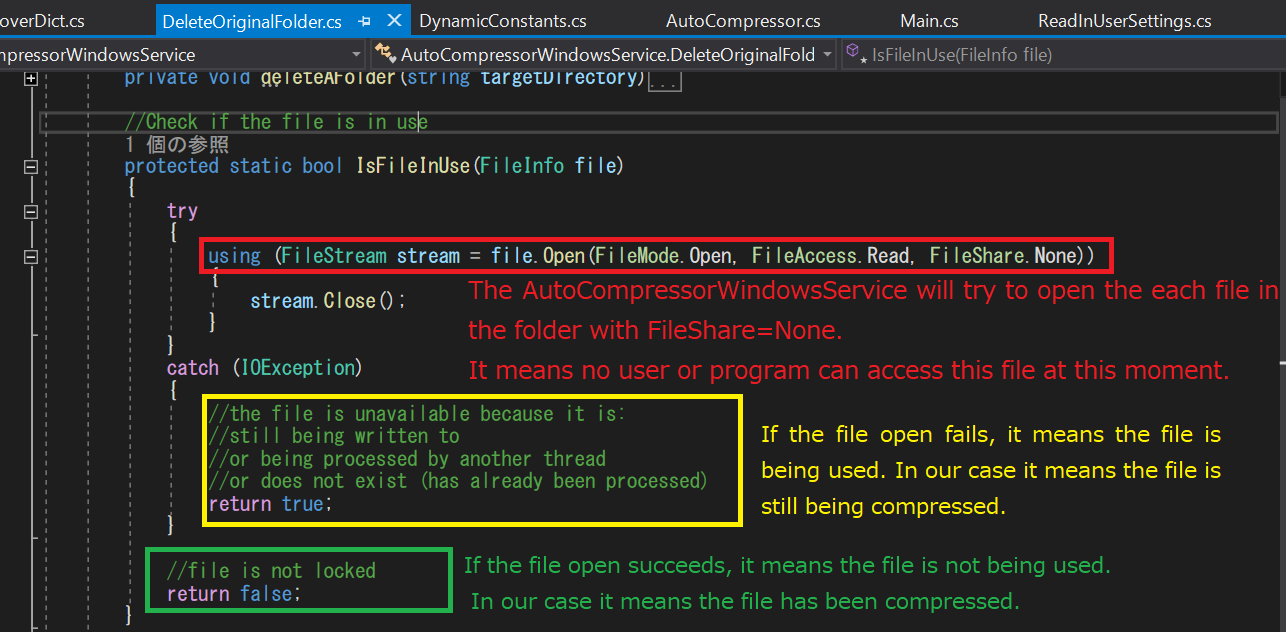
(下記のAutoCompressorWindowsServiceのsource codeのscreenshotの黄色のところ)

If the file open succeeds, it means the file is not being used. In our case it means the file has been compressed.

(下記のAutoCompressorWindowsServiceのsource codeのscreenshotの緑色のところ)

After the AutoCompressorWindowsService has finished trying to open all the files in the folder, which means the folder is not being compressed,

the AutoCompressorWindowsService will delete the folder.



詳細設計：

DeleteOriginalFolder.cs is created for this objective.

DeleteOriginalFolder.cs Class member functions introduction:

Purpose:

Delete the specified folder. (It will wait until the specified folder is no longer in use, and then delete the folder.)

Class member functions:

deleteAfterCompress(string targetDirectory)：

Wait until the specified folder is not in use, and the　delete it.

waitForCompressFinish(string targetDirectory)

Wait for all the files to be not in use.

deleteAFolder(string targetDirectory)

Delete a folder.

IsFileInUse(FileInfo file)

Check if the file is in use.(It tries to open the file with “FileShare=None”, if the file can be opened it means the file is not being used.)

resetDeletionList()

Clear the content of the deletionList

機能2について：

機能2の目的：

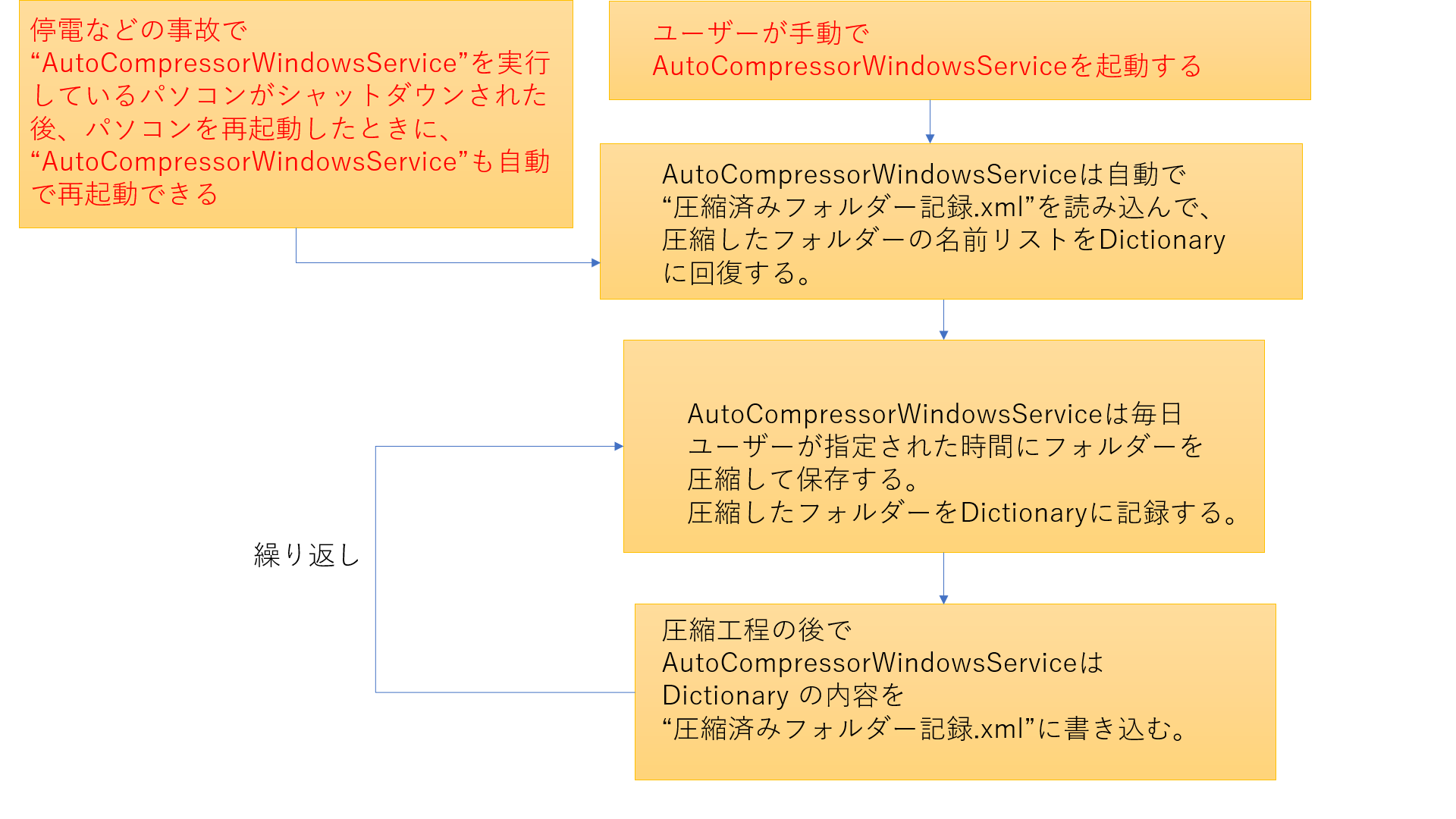
フォルダーを圧縮して保存してから、自動で元のフォルダーを削除する機能を使っている場合、オペレーターが操作を間違って作成した同じ名前のフォルダーが既に存在しているZIPファイルを置き換えないように同じ名前のフォルダーを圧縮して削除しないで、そのまま元のフォルダーの形でNASに保存されています。そして、停電などの事故があっても圧縮済みフォルダー記録を回復できるようにします。

解決策：

Dictionary Data Structureを利用して既に圧縮したフォルダーの名前を記録する。

そして、Dictionary Data Structureの内容を定期的にtxtファイルにoutputしてback upして、AutoCompressorWindowsService（自動圧縮プログラム）は停電で止まった後、再起動する時にまたtxtファイルにback up されている圧縮したフォルダーの名前をDictionary Data Structureに読み込みます。

Only the folder that has no repeated name will be compressed to a ZIP file and deleted.



AutoCompressorWindowsServiceは再起動するときに圧縮記録をDictionary に回復できます：

１．ユーザーが手動でAutoCompressorWindowsService を起動するとき

２．停電などの事故で“AutoCompressorWindowsService”を実行しているパソコンがシャットダウンされた後、パソコンを再起動したとき

AutoCompressorWindowsServiceは自動で“圧縮済みフォルダー記録.json”を読み込んで、圧縮したフォルダーの名前リストをDictionary に回復できます。

圧縮済みフォルダー記録.json：

It records the compressed folder names and their status.

There are 3 status:

1. 圧縮して保存しました。

It means the folder name does not exist in the Dictionary, so it is compressed to a ZIP file, and deleted afterwards.

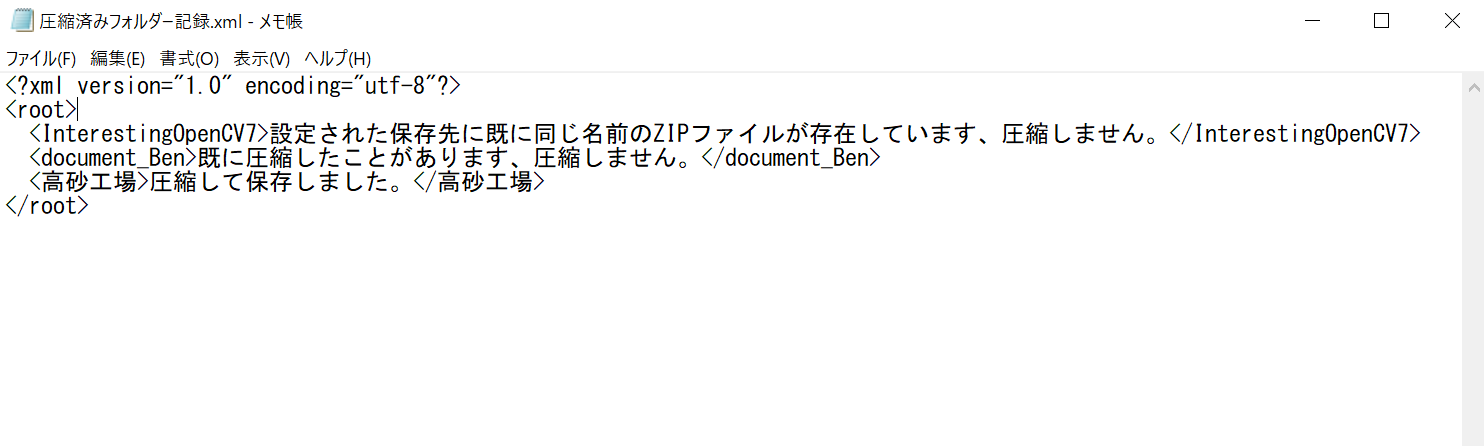
2. 既に圧縮したことがあります、圧縮しません。

It means the folder name has existed in the Dictionary, which means there is a folder with the same name be compressed before. Therefore, the current folder will not be compressed, and will not be deleted.

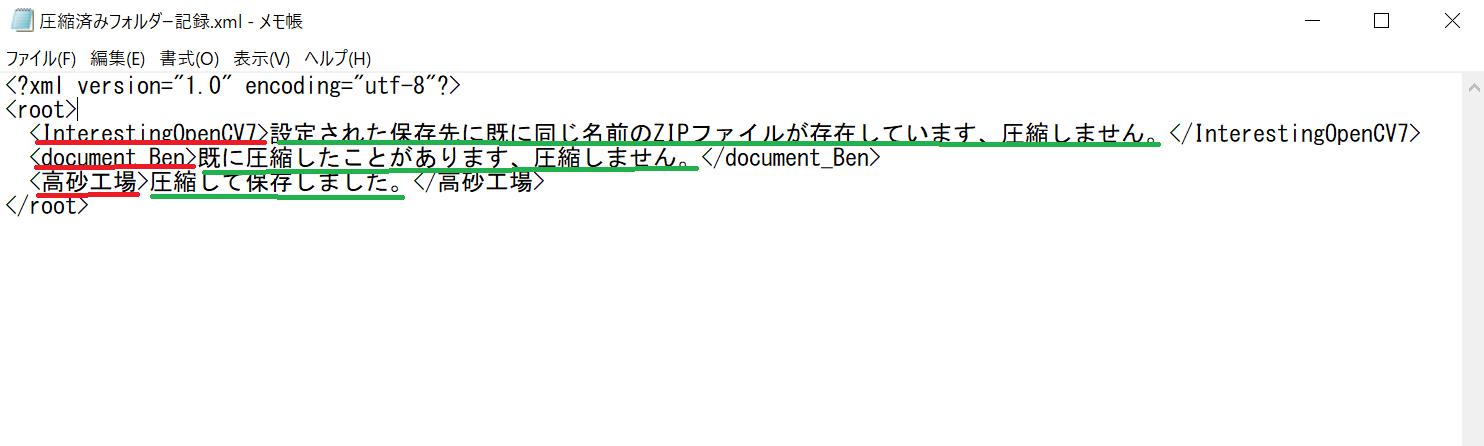
3. 設定された保存先に既に同じ名前のZIPファイルが存在しています、圧縮しません。

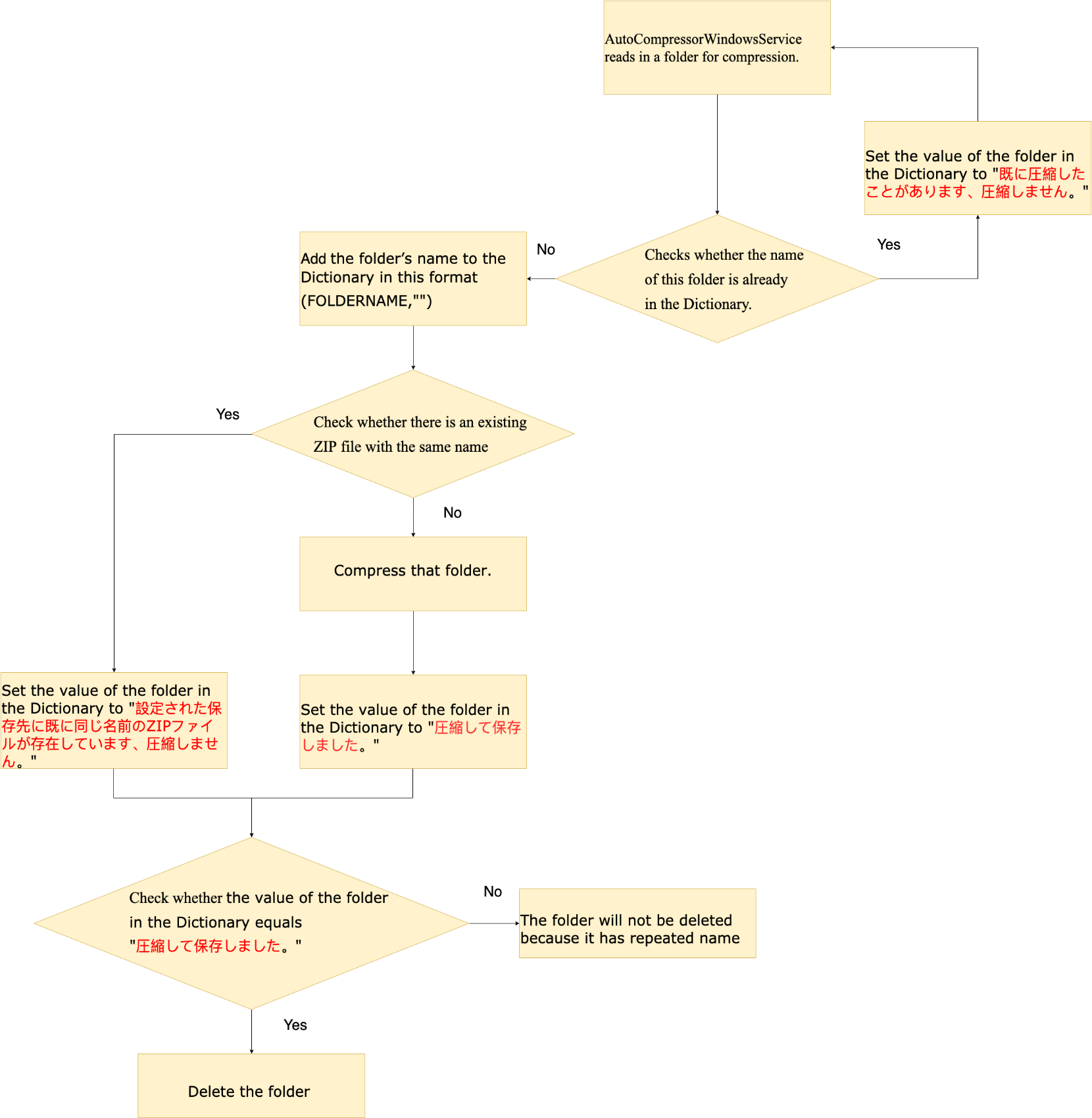
It means the folder name does not exist in the Dictionary, but there is a ZIP file with the same name in the storage destination, so the current folder will not be compressed, and will not be deleted.

圧縮済みフォルダー記録.json might look like this.



The folder name is recorded inside the tag(The red part in the figure below), and its status is recorded between 2 tags(The green part in the figure below).



The following is the overall flowchart:　 Only the folder that has no repeated name will be compressed to a ZIP file and deleted.

詳細設計：

Backup\_RecoverDict.cs is created for this objective.

Backup\_RecoverDict.cs Class member functions introduction:

Purpose:

Recover the content from the 圧縮済みフォルダー記録.json/圧縮済みフォルダー記録.xml to the Dictionary, and back up the content of the Dictionary to the 圧縮済みフォルダー記録.json/圧縮済みフォルダー記録.xml.

Class member functions:

recoverDictFromXMLFile (string XMLSrcPath, Dictionary<string, string> dict)

Recover the content from a xml file to a dictionary.

In our case, we recover the 圧縮済みフォルダーの記録 to a dictionary from a xml file.

backupDictToXMLFile(string xmlStorePath, Dictionary<string, string> dict)

Save the content of the dictionary to a xml file.

In our case, we save the content of the 圧縮済みフォルダーの記録dictionary to a xml file

recoverDictFromJSONFile(string jsonFilePath)

Recover the content from a xml file to a dictionary

In our case, we recover the 圧縮済みフォルダーの記録 to a dictionary from a json file.

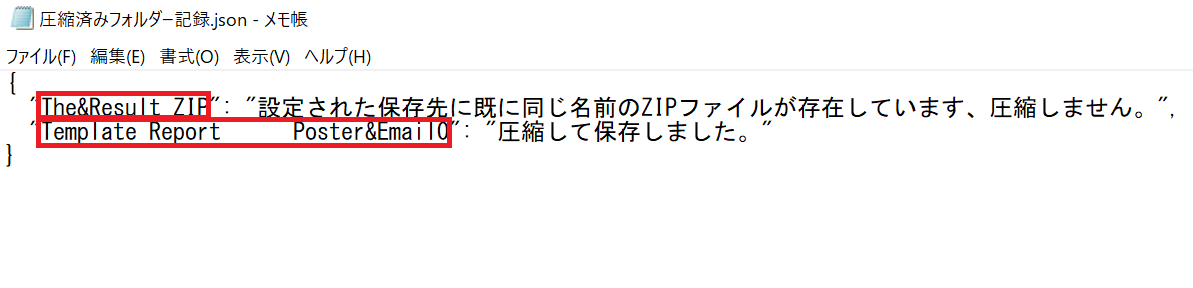
backupDictToJSONFile(string jsonFilePath, Dictionary<string, string> dict)

Save the content of the dictionary to a json file

In our case, we save the content of the 圧縮済みフォルダーの記録dictionary to a json file

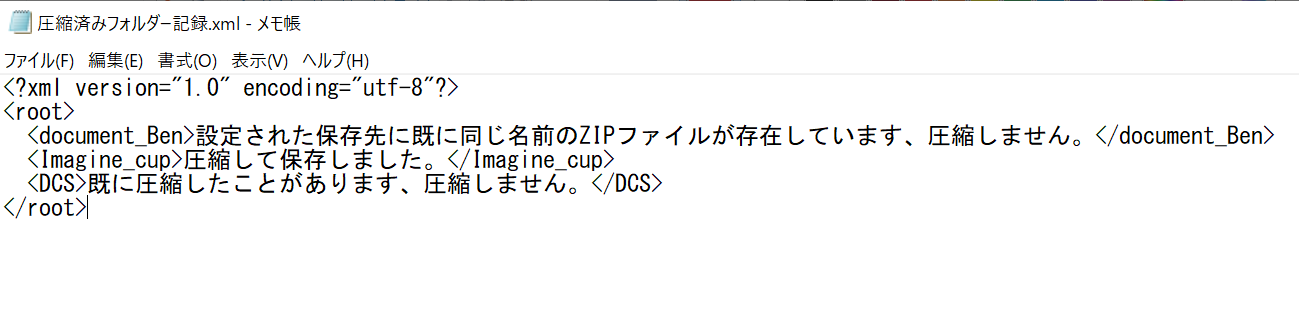
The reason why we use JSON file to back up 圧縮済みフォルダーの記録:

1. JSON file supports structured data format like XML, but the file size is smaller than XML.
2. JSON format allows the folder name to contain white space, “&” while XML format does not support this merit(as shown at the red parts in the screenshot below).



1. Compared to XML, JSON format is much easier for users to check and understand the compression status of each folder(as shown in the following screenshots).
   * Using XML format to back up 圧縮済みフォルダーの記録:

e.g., <folder name>compression status</folder name>



* + Using JSON format to back up 圧縮済みフォルダーの記録:

e.g., ”folder name” : “compression status”

