# Archer ZIP Management Tool & Utility

The Archer ZIP Management Tool & Utility is a custom object that can be configured and placed on an Archer Application or Questionnaire layout. It creates a ZIP file and automatically downloads all files present within specific attachment fields that are either present on that record or across specified cross-referenced records.

On this page

* [Release history](#Releasehistory)
* [Overview of the Archer ZIP Management Tool & Utility](#X13bf86e1647478e0ea49926d293c3f61e530ba7)
  + [Key features and benefits](#Keyfeaturesandbenefits)
  + [Prerequisites (ODA and system requirements)](#PrerequisitesODAandsystemrequirements)
* [Archer ZIP Management Tool and Utility Workflow](#X60d212676397d69a5c935180a0929d100e7e96e)
* [Installing the Archer ZIP Management Tool & Utility Package](#Xbc379660216472157141ddef8bcf4c8fe66b816)
  + [Installation overview](#Installationoverview)
    - [Task 1: Extract the files](#Task1Extractthefiles)
    - [Task 2: Configure the integration](#Task2Configuretheintegration)
    - [Task 3: Test the installation](#Task3Testtheinstallation)
* [Configuring the Archer ZIP Management Tool & Utility](#X7688b294d115ce49124c80c5c8411beb8a0f042)
  + [Custom object configuration](#Customobjectconfiguration)
  + [Create the custom object](#Createthecustomobject)
* [Sample configurations](#Sampleconfigurations)
  + [Download files from a single attachment field on a single record](#X14e02a1bbdb327afc090605fbbf23db5d30c49a)
  + [Download files from multiple attachment fields and cross-referenced records](#X0b95612e54e1b77d8bcb022d6ff4425ee503033)
* [Certification environment](#Certificationenvironment)

## Release history

Last updated: August 2022

## Overview of the Archer ZIP Management Tool & Utility

### Key features and benefits

The Archer ZIP Management tool & utility enables Archer users to:

* Download the files associated with one or more attachment fields present on a single record or within any or all cross-referenced applications into a single ZIP file by clicking a link on a record page layout.
* Eliminate the need to manually click each individual file present within a record to download it to their desktops.
* Eliminate the need to traverse multiple cross-referenced records and find the associated attachment fields to download.
* Automatically create appropriately named subfolders, eliminating the need to manually create folders and move downloaded files into those folders.
* Only download files from records and fields to which they have access.  If users are not permissioned to the record (current or cross-referenced) on which an attachment field resides or if they do not have access to a private attachment field, their download will omit these folders and files.

Benefits include:

* Reduction in time and effort downloading individual attachments
* Automation for organizing downloaded attachments pertaining to specific sets of records
* Allowing users to download files only from records and fields to which they have access

### Prerequisites (ODA and system requirements)

| Components | Prerequisites |
| --- | --- |
| Archer Solution Area(s) | Any |
| Uses Custom Application | No |
| Archer Instance | This offering has been validated on Archer Platform release 6.12 |
| Supported Archer Environments | * On-Premises * Archer Hosted * Archer SaaS |

## Archer ZIP Management Tool and Utility Workflow

The Archer ZIP Management tool & utility provides a custom object which leverages Archer APIs to create a ZIP file containing files present in a configurable list of attachment fields present within a single record and/or related cross-referenced records in a configurable folder structure. Once the ZIP file is generated, it will be saved with a configurable and dynamically generated file name and automatically downloaded to the user’s desktop via their browser.

If the custom object is configured to be called via the click event of a pre-existing element (e.g., an Advanced Workflow User Action Node transition button), its execution ends by calling that element’s original click event (e.g., to handle the workflow action).

## Installing the Archer ZIP Management Tool & Utility Package

### Installation overview

Complete the following tasks to install the offering.

#### Task 1: Extract the files

1. Unzip the contents of the file “Archer\_ZIP\_Management\_Tool&Utility\_6.12.zip” into the target folder.
2. This file contains one custom object script which must be edited using an IDE or text editor.

#### Task 2: Configure the integration

See [Configuring the Archer ZIP Management Tool & Utility](#Xe1e2d04a59ae4d0f079d2158b469b411e96d68b) for complete information.

#### Task 3: Test the installation

Test the application according to your company standards and procedures to ensure that the use case works with your existing processes.

## Configuring the Archer ZIP Management Tool & Utility

### Custom object configuration

Open the attached file DownloadAllAttachmentsCustomObjectwithMultiLevelSupport.js. in an IDE (e.g., Visual Studio, Visual Studio Code) or a text editor (e.g., Notepad++, Notepad) to make edits needed for the specific implementation. The structure and contents of the ZIP file generated by the custom object is controlled via configuration of the objDownloadAllAttachmentsParams object.  It controls the following:

* Creation, naming, and placement of a hyperlink which when clicked will initiate the creation of the ZIP file
* Name of the ZIP file which gets created
* Name of the optional folder within the ZIP where files and/or subfolders will be placed
* List of attachment fields (if any) within the current record whose files will be downloaded into the ZIP file, and if separate subfolders corresponding to each attachment field will be created within the ZIP file to separate the files present in those fields
* List of cross-referenced fields (if any) that the custom object will traverse in order to create subfolders and/or download files from attachment fields specified
  + List of attachment fields (if any) within these cross-referenced records whose files will be downloaded into the ZIP file, and if separate subfolders corresponding to each attachment field will be created within the ZIP file to separate the files present in those fields

Format the objDownloadAllAttachmentsParamsobject by editing the following parameters as per your requirements:

            varobjDownloadAllAttachmentsParams = {

                AddStandaloneDownloadButton: true, //[true|false]–if true, a button will be added to the record layout in the location of the custom object with text equal to the StandaloneDownloadButtonTextvalue

                StandaloneDownloadButtonText: “”, //[string] –text of the button that will be added to the record layout if AddStandaloneDownloadButton = true

               CreateLinkForFunction: true, //[true|false] –iftrue, a download icon link will be added next to the fieldname specified in the FileToAppendDownloadLink

                FieldToAppendDownloadLink: “”, //[string] –the field name to add a Download link to if opting not to render a standalone button.  If no field-adjacent link is to be rendered, set this to “”

                DownloadFileNameField: “”, //[string] –the name of the field whose name or value will be used in the {DownloadFileNameField.Name} or {DownloadFileNameField.Value} parameter of the DownloadFileNameproperty.

                DownloadFileName: “{DownloadFileNameField.Value} – Attachments – {YYYY-MM-DD}.zip”, //[string] –The filename of the .zip file containing all the selected files and folders.  Can include a combination of static text, {DownloadFileNameField.Name}, {DownloadFileNameField.Value}, and date parameters {YYYY-MM-DD} or {YYYYMMDDHHMMSS}.  Must end in .zip.

                AttachmentFields: [], //[string array] –an array (comma-separated list of quote-delimited string values within []) consisting of one or more Attachment Fieldnames within this level from which attachments are to be downloaded

                FolderField: “”, //[string]–the name of the field whose name or value will be used in the {FolderField.Name} or {FolderField.Value} parameters passed to the FolderNameproperty

                FolderName: “{FolderField.Value}”, //[string] –the name of the folder that will be created inside the .zip file to store the files in the AttachmentFieldsarray

                CreateSubfolderForEachAttachmentField: true, //[true|false] –if true, a subfolder will be created inside the FolderName folder for each AttachmentFields value, with the files from that specific field being placed inside.  The folder name will be set to the AttachementFields fieldname.

                DownstreamAttachmentParameters: [{

                    AttachmentFields: [],

                    DownstreamXRefFieldNameOnParent: “”,

                    FolderField: “”,

                    FolderName: “{FolderField.Value}”,

                    CreateSubfolderForEachAttachmentField: false,

                    DownstreamAttachmentParameters: [{

                        DownstreamXRefFieldNameOnParent: “”,

                        FolderField: “”,

                        FolderName: “{FolderField.Value}”,

                        AttachmentFields: [],

                        CreateSubfolderForEachAttachmentField: false,

                        DownstreamAttachmentParameters: []

                    }]

                }] //[object array] –An array of one or more objects (each object is denoted by {} – additional objects are comma separated) that defines nested cross-referenced fields from which attachments can be downloaded.  If you had two cross-referenced fields from which you wanted to pull attachments (e.g., Remediation Plans and Exception Requests from a parent Findings record), you would add two objects in the DownstreamAttachmentParameters array.  The DownstreamXRefFieldNameOnParent denotes the field name on the parent level/module that stores the cross-referenced records.  The DownstreamAttachmentParameters property stores an array of downstream objects should you need to go down to child cross-referenced levels/modules.  If no attachments from downstream levels/modules need to be brought in, set this to an empty array [].

            };

If you wish the downloadAllAttachments function to run before and in conjunction with an existing button click (save, apply, or a workflow action node), add as many of the following wireUpFunctionToElementEventByText or wireUpFunctionToElementEventByAttribute function calls as needed to match the user action buttons that when clicked will call the downloadandZipAllAttachments function.

The second/third arguments of the respectivewireUpFunctionToElementEventByText/wireUpFunctionToElementEventByAttribute calls needs to match the exact text of the user action button or calling element.  If the CreateLinkForFunction property is set to true, a downloadAllAttachments hyperlink will be automatically inserted next to the Repository Cross-Reference or Attachment Field on this application/questionnaire.

Example wireUpFunctionToElementEventByText/wireUpFunctionToElementEventByAttribute calls:

downloadAllAttachmentsCustomObject.wireUpFunctionToElementEventByText("a", "Close Exam", "onclick", downloadAllAttachmentsCustomObject.functionString, false);//Would run before executing an AWF User Action Node transition named "Close Exam".

downloadAllAttachmentsCustomObject.wireUpFunctionToElementEventByAttribute("a", "id", "master\_btnApply1", "onclick", downloadAllAttachmentsCustomObject.functionString, false);//Would run when the user clicks the SAVE button

### Create the custom object

* Add a custom object to the layout named“DownloadAllAttachments” or something similar.  Please note that the custom object must be placed outside of any Tab Set if configured to run before a User Action Transition or within the Tab Set that the field specified in the objDownloadAllAttachmentsParams.FieldToAppendDownloadLink resides in (if the objDownloadAllAttachmentsParams.AddStandaloneDownloadButtonproperty is set to true) to ensure the script executes when the record page loads.
* Copy and paste the contents of the modifiedDownloadAllAttachmentsCustomObjectwithMultiLevelSupport.js file into the Code field.
* Select Display the custom object when viewing or editing a record.

## Sample configurations

### Download files from a single attachment field on a single record

Below is an example of anobjDownloadAllAttachmentsParamsconfiguration for simply downloading the contents of the Attachment field within the Findings application with no additional attachments from any cross-reference records being downloaded.

* It will add a button on the layout named “DOWNLOAD ALL FINDING ATTACHMENTS” as well as add a download hyperlink next to the “Attachment” field.  (In a real-world example, one would select to display either the button or the link.)
* The filename of the ZIP file produced by clicking the button or the link will be FND-{DownloadFileNameField.Value} - Attachments - {YYYY-MM-DD}.zip where {DownloadFileNameField.Value} will be replaced by the value of the Finding ID field of the record from which the custom object is executed and {YYYY-MM-DD} will be the date the custom object is run in YYYY-MM-DD format.
* A main folder will be created inside the ZIP file whose name will be set to the value of the Finding ID field of the record from which the custom object is executed.
  + The files present in the Attachment field will be placed inside this main folder.

The objDownloadAllAttachmentsParams object would be set as follows:

varobjDownloadAllAttachmentsParams = {

AddStandaloneDownloadButton: true, //if true, a button will be added to the record layout in the location of the custom object with text equal to the StandaloneDownloadButtonTextvalue.

StandaloneDownloadButtonText: "DOWNLOAD ALL FINDING ATTACHMENTS", //text of the button that will be added to the record layout if AddStandaloneDownloadButton = true

CreateLinkForFunction: false,

FieldToAppendDownloadLink: "Attachment", //the field name to add a Download link to if opting not to render a standalone button.  If no field adjacent link is to be rendered, set this to ""

DownloadFileNameField: "Finding ID", //the name of the field whose name or value will be used in the {DownloadFileNameField.Name} or {DownloadFileNameField.Value} parameter of the DownloadFileNameproperty

DownloadFileName: "FND-{DownloadFileNameField.Value} - Attachments - {YYYY-MM-DD}.zip", //The filename of the .zip file containing all of the selected files and folders.  Can include a combination of static text, {DownloadFileNameField.Name}, {DownloadFileNameField.Value}, and date parameters {YYYY-MM-DD} or {YYYYMMDDHHMMSS}.  Must end in .zip.

AttachmentFields: ["Attachment"], //an array (comma separated list of values within []) of one or more Attachment Fieldnames within this level from which attachments are to be downloaded.

FolderField: "Finding ID", //the name of the field whose name or value will be used in the {FolderField.Name} or {FolderField.Value} parameters passed to the FolderNameproperty.

FolderName: "FND-{FolderField.Value}", //the name of the folder that will be created inside the .zip file to store the files in the AttachmentFields array

CreateSubfolderForEachAttachmentField: false, //if true, a subfolder will be created inside the FolderName folder for each AttachmentFields value, with the files from that specific field being placed inside.  The folder name will be set to the AttachementFields fieldname.

DownstreamAttachmentParameters: [] //An array of one or more objects (each object is denoted by {} -- additional objects are comma separated) that defines nested cross-referenced fields from which attachments can be downloaded.  If you had two cross-referenced fields from which you wanted to pull attachments (e.g. Remediation Plans and Exception Requests from a parent Findings record), you'd add two objects in the DownstreamAttachmentParameters array.  The DownstreamXRefFieldNameOnParent denotes the field name on the parent level/module that stores the cross-referenced records.  The DownstreamAttachmentParameters property stores an array of downstream object should you need to go down to child cross-referenced levels/modules.  If no attachments from downstream levels/modules need to be brought in, set this to an empty array [].

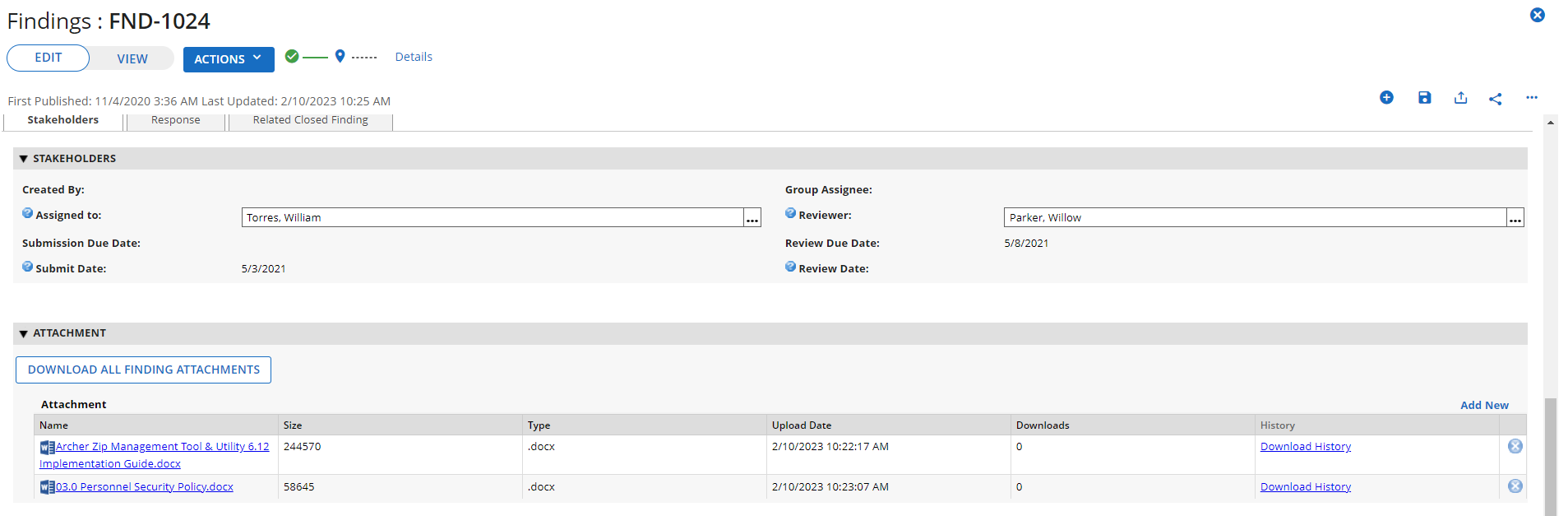
            };

downloadAllAttachmentsCustomObject = new DownloadAllAttachmentsCustomObject(objDownloadAllAttachmentsParams);

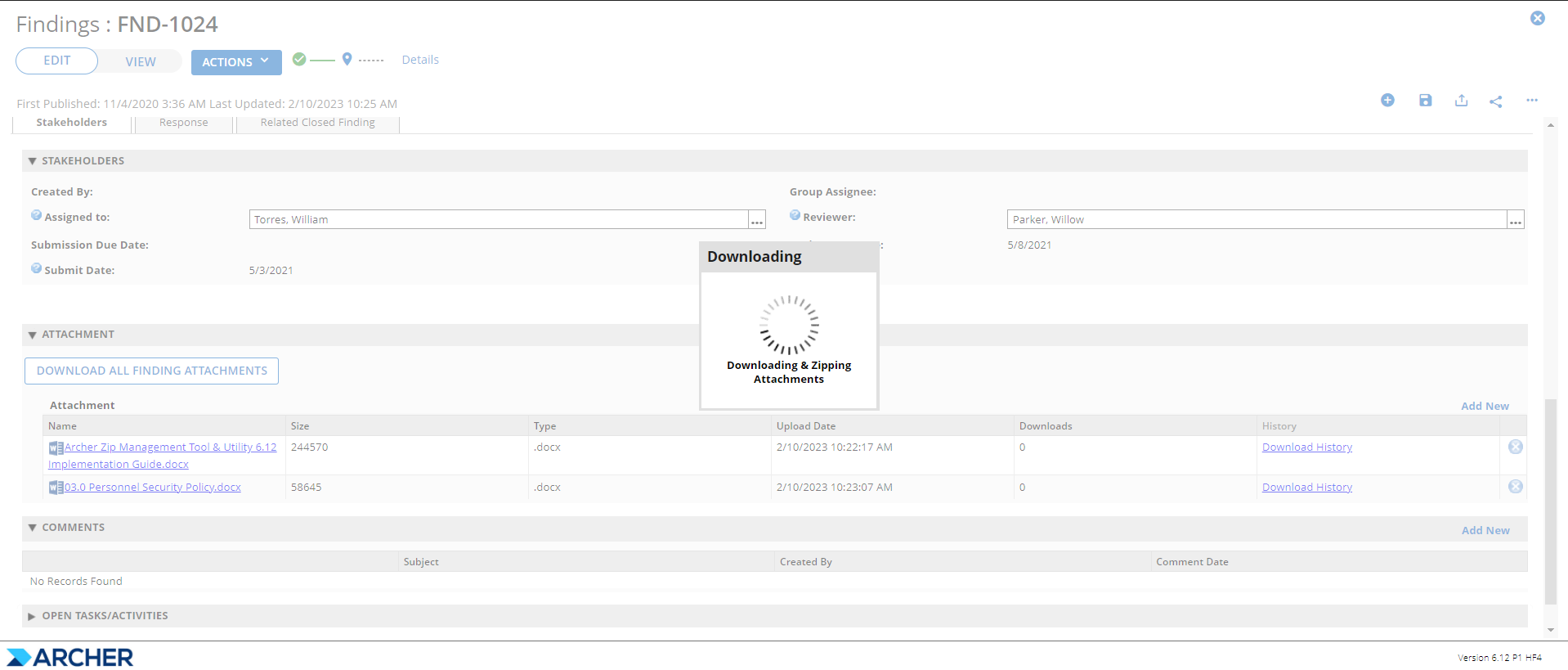
        };

Based on the configuration above, the custom object code will display the following layout.  Note that both a DOWNLOAD ALL FINDING ATTACHMENTS button as well as a download link to the right of the Attachment field were added.  To remove the button or the link, set the AddStandaloneDownloadButtonor the CreateLinkForFunctionpropertiesto false, respectively.

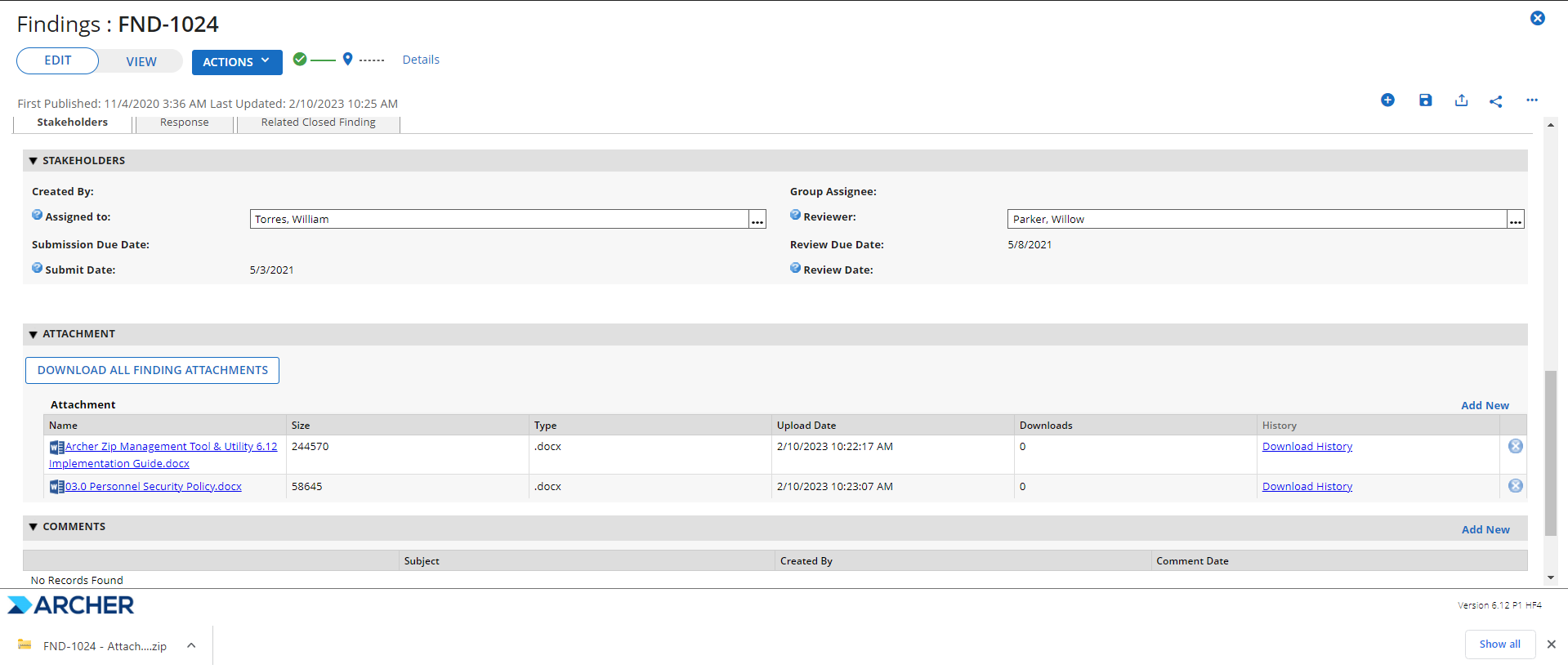
Based on the configuration above, clicking the button or the link on a Findings record named “FND-1024” with an attachment field named “Attachment” containing the two Word documents seen above would produce views and a ZIP file.

Before clicking the DOWNLOAD ALL FINDING ATTACHMENTS button:

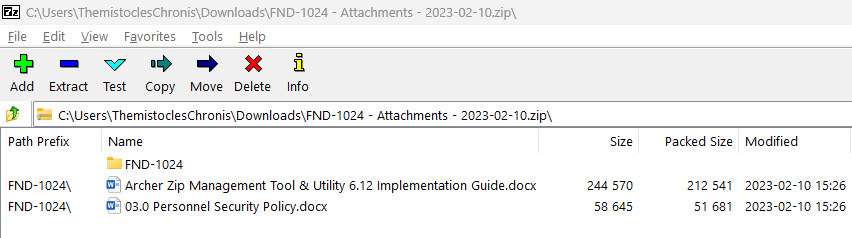
After clicking the DOWNLOAD ALL FINDING ATTACHMENTS button:



After the function completes:



The contents of the FND-1024 - Attachments - 2023-02-10.zip file:



### Download files from multiple attachment fields and cross-referenced records

Below is an example ofan objDownloadAllAttachmentsParams configuration for downloading the Audit Snapshot,Audit Engagement within an Audit Engagement along with any Supporting Documents present within the Audit Workpapers: Audit Procedures records cross-referenced to the Audit Workpapers: Audit Programrecords cross-referenced to the current Audit Engagement record.

* It will add a button on the layout named “DOWNLOAD ALL ENGAGEMENT & WORKPAPER ATTACHMENTS”.
* The filename of the ZIP file produced by clicking the button or the link will be {DownloadFileNameField.Value} - Attachments - {YYYY-MM-DD}.zip where {DownloadFileNameField.Value} will be replaced by the value of the Engagement Name field of the record from which the custom object is executed and {YYYY-MM-DD} will be the date the custom object is run in YYYY-MM-DD format.
* A main folder will be created inside the ZIP file whose name will be set to the value of the Engagement Name field of the record from which the custom object is executed.
  + Two subfolders named “Engagement Snapshot” and “Final Audit Report”will be added to the main folder and will store the file(s) attached to the Engagement Snapshot and Final Audit Report fields present on the record from which the custom object is executed.
  + Additional subfolders below the main folder will be created for each Audit Workpapers: Audit Program record present in the Audit Workpapers cross-reference present on the record from which the custom object is executed.  The names of these subfolders will correspond to the Program Name field present on each of these Audit Workpapers.  (Note, that no attachment fields within the cross-referenced Audit Workpapers: Audit Program records will actually be downloaded; these records are only used to create subfolders and as a link to child Audit Workpapers: Audit Procedures records from which attachments will be downloaded.)
    - Subfolders below each Audit Program folder will be created for each Audit Workpapers: Audit Procedures record present in the Audit Procedures cross-reference present within each Audit Program record.  The names of these subfolders will correspond to the Procedure Name field present on each of these Audit Procedures.  These subfolders will store the files present in the Supporting Documents field present on each respective Audit Procedures record.
  + Please note that folders will only appear inside the resulting ZIP file if at least one file is present in it or any of its subfolders (i.e., no empty folders will be created).

The objDownloadAllAttachmentsParams object would be set as follows:

            varobjDownloadAllAttachmentsParams = {

                AddStandaloneDownloadButton: true, //if true, a button will be added to the record layout in the location of the custom object with text equal to the StandaloneDownloadButtonTextvalue

                StandaloneDownloadButtonText: "DOWNLOAD ALL ENGAGEMENT & WORKPAPER ATTACHMENTS", //text of the button that will be added to the record layout if AddStandaloneDownloadButton = true

               CreateLinkForFunction: false, //if true, a download icon link will be added next to the fieldname specified in the FileToAppendDownloadLink

                FieldToAppendDownloadLink: "Engagement Snapshot", //the field name to add a Download link to if opting not to render a standalone button.  If no field adjacent link is to be rendered, set this to ""

                DownloadFileNameField: "Engagement Name", //the name of the field whose name or value will be used in the {DownloadFileNameField.Name} or {DownloadFileNameField.Value} parameter of the DownloadFileNameproperty.

                DownloadFileName: "{DownloadFileNameField.Value} - Attachments - {YYYY-MM-DD}.zip", //The filename of the .zip file containing all the selected files and folders.  Can include a combination of static text, {DownloadFileNameField.Name}, {DownloadFileNameField.Value}, and date parameters {YYYY-MM-DD} or {YYYYMMDDHHMMSS}.  Must end in .zip.

                AttachmentFields: ["Engagement Snapshot", "Final Audit Report"], //an array (comma separated list of values within []) of one or more Attachment Fieldnames within this level from which attachments are to be downloaded.

                FolderField: "Engagement Name", //the name of the field whose name or value will be used in the {FolderField.Name} or {FolderField.Value} parameters passed to the FolderNameproperty.

                FolderName: "{FolderField.Value}", //the name of the folder that will be created inside the .zip file to store the files in the AttachmentFieldsarray.

                CreateSubfolderForEachAttachmentField: true, //if true, a subfolder will be created inside the FolderName folder for each AttachmentFields value, with the files from that specific field being placed inside.  The folder name will be set to the AttachementFields fieldname.

                DownstreamAttachmentParameters: [{

                    AttachmentFields: [],

                    DownstreamXRefFieldNameOnParent: "Audit Workpapers",

                    FolderField: "Program Name",

                    FolderName: "{FolderField.Value}",

                    CreateSubfolderForEachAttachmentField: false,

                    DownstreamAttachmentParameters: [{

                        DownstreamXRefFieldNameOnParent: "Audit Procedures",

                        FolderField: "Procedure Name",

                        FolderName: "{FolderField.Value}",

                        AttachmentFields: ["Supporting Documents"],

                        CreateSubfolderForEachAttachmentField: false,

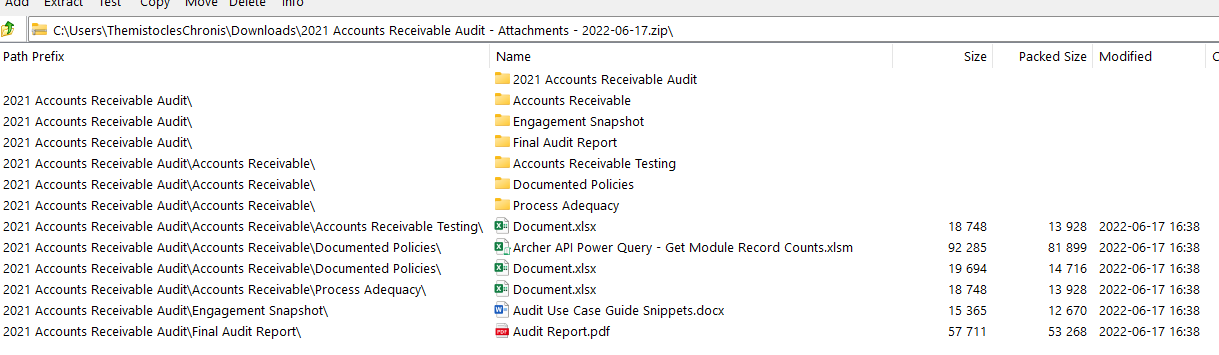
                        DownstreamAttachmentParameters: []

                    }]

                }] //An array of one or more objects (each object is denoted by {} -- additional objects are comma separated) that defines nested cross-referenced fields from which attachments can be downloaded.  If you had two cross-referenced fields from which you wanted to pull attachments (e.g., Remediation Plans and Exception Requests from a parent Findings record), you'd add two objects in the DownstreamAttachmentParameters array.  The DownstreamXRefFieldNameOnParent denotes the field name on the parent level/module that stores the cross-referenced records.  The DownstreamAttachmentParameters property stores an array of downstream object should you need to go down to child cross-referenced levels/modules.  If no attachments from downstream levels/modules need to be brought in, set this to an empty array [].

            };

Based on the configuration above, the custom object code executed from an Audit Engagement named “2021 Accounts Receivable Audit” with Audit Workpapers named “Accounts Receivable” and Audit Procedures cross-referenced to the “Accounts Receivable” Audit Workpaper named “Accounts Receivable Testing”, “Documented Policies”, and “Process Adequacy” would produce aZIP filelike the screenshot below:



## Certification environment

Date tested: January 2023

| Product Name | Version Information | Operating System |
| --- | --- | --- |
| Archer | 6.12 | Windows 2019 |