

User Guide: fn_utilities_v2.0.0

Table of Contents

- [App Host Setup](#)
- [Integration Server Setup](#)
- [Function - Utilities: Attachment Hash](#)
- [Function - Utilities: Attachment to Base64](#)
- [Function - Utilities: Attachment Zip Extract](#)
- [Function - Utilities: Attachment Zip List](#)
- [Function - Utilities: Base64 to Artifact](#)
- [Function - Utilities: Base64 to Attachment](#)
- [Function - Utilities: Call REST API](#)
- [Function - Utilities: Domain Distance](#)
- [Function - Utilities: Email Parse](#)
- [Function - Utilities: Excel Query](#)
- [Function - Utilities: Expand URL](#)
- [Function - Utilities: Extract SSL Cert From Url](#)
- [Function - Utilities: Get Contact Info](#)
- [Function - Utilities: JSON2HTML](#)
- [Function - Utilities: Parse SSL Certificate](#)
- [Function - Utilities: PDFid](#)
- [Function - Utilities: Resilient Search](#)
- [Function - Utilities: Shell Command](#)
- [Function - Utilities: String to Attachment](#)
- [Function - Utilities: Timer](#)
- [Function - Utilities: XML Transformation](#)
- [Rules](#)

Release History

| Version | Date | Notes |
|---------|--------|--|
| 2.0.0 | 7/2020 | Numerous fixes, improved Rules and workflows and only Python 3 supported |
| 1.0.15 | 5/2020 | Bug fixes |
| 1.0.14 | 5/2020 | Shell Command support for Remote Linux Execution |

App Host Setup

This app is available for use in App Host.

When using Shell Command, several Linux commands have been installed on the container including: dig, nslookup, traceroute and whois. These commands can be specified within the app.config file directly such as: `nslookup=nslookup ""`. Other commands not loaded within the container can be accessed via remote shell execution. See the section [Function - Utilities: Shell Command](#) for more information.

Integration Server Setup

Note: this version of fn_utilities will only run in a Python 3 environment. This is due to changes in dependent python packages with Python 2's end of life. If you continue to use Python 2 in your Integration Server environment, use previous versions of this app.

Function - Utilities: Attachment Hash

Calculate hashes for a file attachment. Returns **md5**, **sha1**, **sha256** and other hashes of the file content. Those hashes can then be used as artifacts or in other parts of your workflows.

The screenshot shows the Resilient interface with the 'Customization Settings' for a workflow function named 'Example: Attachment Hash'. The interface includes a top navigation bar with 'Resilient' logo and tabs for 'Dashboards', 'Simulations', 'Incidents', and 'Create'. The 'Create' tab is active. Below the navigation bar, there are tabs for 'Layouts', 'Rules', 'Scripts', 'Workflows', 'Functions', 'Message Destinations', 'Phases & Tasks', 'Incident Types', 'Breach', and 'Artifacts'. The 'Workflows' tab is selected, and the sub-tab 'Example: Attachment Hash' is active. On the right, there are buttons for 'Cancel', 'Save & Close', and 'Save'. The main form contains fields for 'Name' (Example: Attachment Hash), 'API Name' (example_attachment_hash), 'Description' (An example that calculates hash artifacts from an attachment.), and 'Object Type' (Attachment). On the right side of the form, there is a table with metadata: 'Creator' (Orchestration Engine), 'Last Modified' (07/17/2019 14:20), 'Last Modified By' (Orchestration Engine), and 'Associated Rules' (Example: Attachment Hash). Below the form, there is a visual workflow diagram showing a sequence of steps: a start node (circle) followed by a function node labeled 'Utilities: Attachment Hash' (with a 'fn' icon), and an end node (circle). A callout box points to the function node with the text: 'Calculate hashes of a file attachment. The results are added to the incident as new artifacts.'

- Inputs:
- Outputs:
- Example Pre-Process Script:
- Example Post-Process Script:

Function - Utilities: Attachment to Base64

Reads a file attachment in the incident, and produces a base64-encoded string with the file attachment content. This content can then be used in combination with other workflow functions to create an artifact, a new file attachment, or to analyze the contents using various tools.

Customization Settings

Layouts Rules Scripts **Workflows** Functions Message Destinations Phases & Tasks Incident Types Breach Artifacts

Workflows / Example: Attachment to Base64



Cancel

Save & Close

Save

Name * Example: Attachment to Base64

API Name * example_attachment_to_base64

Description Convert an Attachment of any type to a Base64 Encoded string

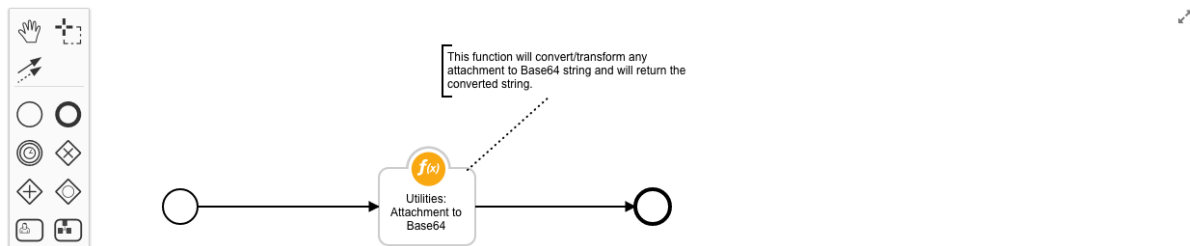
Object Type * Attachment

Creator Orchestration Engine

Last Modified 07/17/2019 14:20

Last Modified By Orchestration Engine

Associated Rules Example: Attachment to Base64



- ▶ Inputs:
- ▶ Outputs:
- ▶ Example Pre-Process Script:
- ▶ Example Post-Process Script:

Function - Utilities: Attachment Zip Extract

Extracts a file from a ZIP file attachment, producing a base64 string.

That string can then be used as input to subsequent functions that might write it as a file attachment, such as a malware sample artifact.

Customization Settings

Layouts Rules Scripts **Workflows** Functions Message Destinations Phases & Tasks Incident Types Breach Artifacts

Workflows / Example: Zip Extract



Cancel

Save & Close

Save

Name * Example: Zip Extract

API Name * example_zip_to_artifact

Description An example showing how to extract a file from a ZIP file attachment.

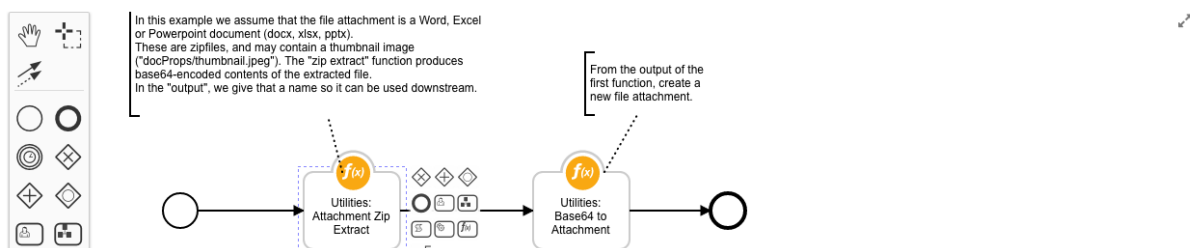
Object Type * Attachment

Creator Orchestration Engine

Last Modified 07/17/2019 14:20

Last Modified By Orchestration Engine

Associated Rules Example: Zip Extract



- ▶ Inputs:
- ▶ Outputs:
- ▶ Example Pre-Process Script:
- ▶ Example Post-Process Script:

Function - Utilities: Attachment Zip List

Reads a ZIP file and produces a list of the compressed files, and a list with detailed information about each file.

Note: The contents of password protected Excel spreadsheets cannot be listed.

Resilient Dashboards Simulations Incidents **Create** Admin User Test Organization

Customization Settings

Layouts Rules Scripts **Workflows** Functions Message Destinations Phases & Tasks Incident Types Breach Artifacts

Workflows / Example: Zip List

Name *Example: Zip List

API Name *example_zip_list

DescriptionAn example showing how to list the contents of a ZIP file attachment.

Object Type *Attachment

CreatorOrchestration Engine

Last Modified07/17/2019 14:20

Last Modified ByOrchestration Engine

Associated RulesExample: Zip List

Function reads the attachment (by id) then produces a list of its contents, in a structured data format. The post-processing script writes these results into a note on the incident.

Utilities: Attachment Zip List

- Inputs:
- Outputs:
- Example Pre-Process Script:
- Example Post-Process Script:

Function - Utilities: Base64 to Artifact

Creates a new artifact from a Base64 string. You can also specify the artifact type and description.

Resilient Dashboards Simulations Incidents **Create** Admin User Test Organization

Customization Settings

Layouts Rules Scripts Workflows **Functions** Message Destinations Phases & Tasks Incident Types Breach Artifacts

Functions / utilities_base64_to_artifact

Name *Utilities: Base64 to Artifact

API Name *utilities_base64_to_artifact

Message Destination *fn_utilities

DescriptionCreate a new artifact from a Base64 string

CreatorOrchestration Engine

Last Modified07/19/2019 16:17

Last Modified ByOrchestration Engine

Associated WorkflowsFunction is not currently referenced by any workflow.

Inputs

base64content

incident_id

artifact_file_type

file_name

content_type

Input Fields

Search...

artifact_file_type

artifact_id

attachment_id

attachment_name

- Inputs:
- Outputs:
- Example Pre-Process Script:
- Example Post-Process Script:

Function - Utilities: Base64 to Attachment

Creates a new attachment from a base64 string. You can also specify the file name and content type to use.

Customization Settings

Layouts Rules Scripts Workflows **Functions** Message Destinations Phases & Tasks Incident Types Breach

Artifacts

Functions / utilities_base64_to_attachment

Name * Utilities: Base64 to Attachment

API Name * utilities_base64_to_attachment

Message Destination * fn_utilities

Description Create a new attachment from a base64 string.

Creator Orchestration Engine

Last Modified 07/19/2019 16:17

Last Modified By Orchestration Engine

Associated Workflows Example: Zip Extract

Inputs

- base64content
- incident_id
- task_id
- file_name
- content_type

Input Fields

- artifact_file_type
- artifact_id
- attachment_id
- attachment_name

- Inputs:
- Outputs:
- Example Pre-Process Script:
- Example Post-Process Script:

Function - Utilities: Call REST API

This function calls a REST web service. It supports the standard REST methods: GET, HEAD, POST, PUT, DELETE and OPTIONS.

The function parameters determine the type of call, the URL, and optionally the headers, cookies and body. The results include the text or structured (JSON) result from the web service, and additional information including the elapsed time.

Customization Settings

Layouts

Rules

Scripts

Workflows

Functions

Message Destinations

Phases & Tasks

Incident Types

Breach

Artifacts

Workflows / Example: Call REST API



Cancel

Save & Close

Save

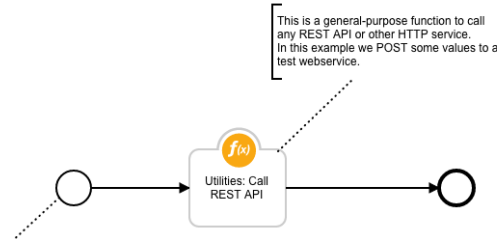
Name * Example: Call REST API

API Name * example_call_rest_api

Description An example of calling a REST API webservice from a workflow.

Object Type * Artifact

Creator Orchestration Engine
Last Modified 07/17/2019 14:20
Last Modified By Orchestration Engine
Associated Rules Example: Call REST API



- Inputs:
- Outputs:
- Example Pre-Process Script:
- Example Post-Process Script:

Function - Utilities: Domain Distance

Identifies similarity between a suspicious domain name and a list of valid domain names. Low distance result indicates a possible spoof attempt. For example, www.ibm.com and www.1bm.com would have a low distance. This can be used for URLs, DNS names and email addresses.

Customization Settings

Layouts

Rules

Scripts

Workflows

Functions

Message Destinations

Phases & Tasks

Incident Types

Breach

Artifacts

Workflows / Example: Domain Distance



Cancel

Save & Close

Save

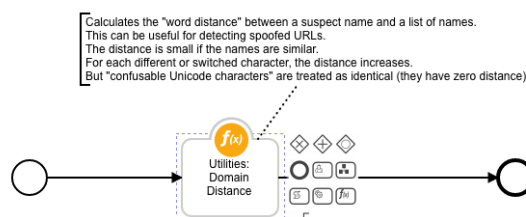
Name * Example: Domain Distance

API Name * example_domain_distance

Description An example testing for confusable domain names

Object Type * Artifact

Creator Orchestration Engine
Last Modified 07/17/2019 14:20
Last Modified By Orchestration Engine
Associated Rules Example: Domain Distance



- Inputs:
- Outputs:
- Example Pre-Process Script:
- Example Post-Process Script:

Function - Utilities: Email Parse

Extracts message headers and parts of the message body from an email message (.eml or .msg).

Any attachments found are added to the incident as artifacts if `utilities_parse_email_attachments` is set to True.

Customization Settings

Layouts Rules Scripts **Workflows** Functions Message Destinations Phases & Tasks Incident Types Breach Artifacts

Workflows / Example: Email Parsing (Attachment)

Name * Example: Email Parsing (Attachment)

API Name * example_email_parsing_attachment

Description Example Workflow showing to parse an Email File (.eml or .msg) from Incident/Task Attachments. Sender and recipient email addresses are added as Artifacts. URLs and IPs found in the email headers or body are also added as Artifacts. The body of the email is added as a Note to the Incident. If attachments are found in the parsed email message, they are added as Email Attachment Artifacts.

Object Type * Attachment

Creator Admin User
Last Modified 07/17/2019 14:20
Last Modified By Orchestration Engine
Associated Rules Example: Email Parsing (Attachment)

Cancel Save & Close Save

Workflow Diagram: Start Node → Utilities: Email Parse → End Node

Supporting Outlook .msg files

- This function relies on `mail-parser>=3.9.3`.

For Integrations Servers:

- To support parsing of Outlook email files (.msg), you need to install the `msgconvert` tool.
- `msgconvert` is a tool written in Perl and can be found in `Email::Outlook::Message` (Centos/RHEL).
- See <https://github.com/SpamScope/mail-parser> for more information on the packaged used.

Install `msgconvert` on CentOS/RHEL based systems:

```
$ sudo yum install cpan
$ sudo cpan -fTi install Email::Outlook::Message
```

For App Host Environments:

- The packages required to parse Outlook .msg files is built into the container.

► Inputs:

► Outputs:

► Example Pre-Process Script:

► Example Post-Process Script:

Function - Utilities: Excel Query

Extracts ranges of data or named ranges specified by the user from a Microsoft Excel document.

The function uses a Python library called `openpyxl` (<http://openpyxl.readthedocs.io/en/stable/>) to interface with Excel files.

Customization Settings

[Layouts](#)
[Rules](#)
[Scripts](#)
[Workflows](#)
[Functions](#)
[Message Destinations](#)
[Phases & Tasks](#)
[Incident Types](#)
[Breach](#)
[Artifacts](#)

Workflows / Example: Create Artifacts From Excel Data

[Cancel](#)
[Save & Close](#)
[Save](#)

| | |
|---------------|---|
| Name * | Example: Create Artifacts From Excel Data |
| API Name * | example_create_artifacts_from_excel_data |
| Description | Created artifacts with information extracted from an excel sheet. |
| Object Type * | Attachment |

Creator Orchestration Engine
Last Modified 07/17/2019 14:20
Last Modified By Orchestration Engine
Associated Rules [Example: Use Excel Data](#)



- ▶ Inputs:
- ▶ Outputs:
- ▶ Example Pre-Process Script:
- ▶ Example Post-Process Script:

Function - Utilities: Expand URL

Takes a shortened URL and follows it through redirects as it expands. The results include each URL, which are added to a new artifact.

Customization Settings

[Layouts](#)
[Rules](#)
[Scripts](#)
[Workflows](#)
[Functions](#)
[Message Destinations](#)
[Phases & Tasks](#)
[Incident Types](#)
[Breach](#)
[Artifacts](#)

Workflows / Example: Expand URL

[Cancel](#)
[Save & Close](#)
[Save](#)

| | |
|---------------|--|
| Name * | Example: Expand URL |
| API Name * | utilities_expand_url |
| Description | Take a url (mostly shortened) and follow it through redirects as it's expanded |
| Object Type * | Artifact |

Creator Orchestration Engine
Last Modified 07/17/2019 14:20
Last Modified By Orchestration Engine
Associated Rules [Example: Expand URL](#)



- ▶ Inputs:
- ▶ Outputs:
- ▶ Example Pre-Process Script:
- ▶ Example Post-Process Script:

Function - Utilities: Extract SSL Cert From Url

This function takes in a HTTPS URL or DNS input, establishes a connection and then attempts to acquire the SSL certificate. If successful, the function then saves the certificate as an artifact of type 'X509 Certificate File'. Works on most URLs including those with self-signed or expired certificates.

The output of this function is a string representation of the certificate saved in PEM format.

Resilient Dashboards Simulations Incidents **Create** Admin User Test Organization

Customization Settings

Layouts Rules Scripts **Workflows** Functions Message Destinations Phases & Tasks Incident Types Breach Artifacts

Workflows / Example: Extract SSL Cert from URL Cancel Save & Close Save

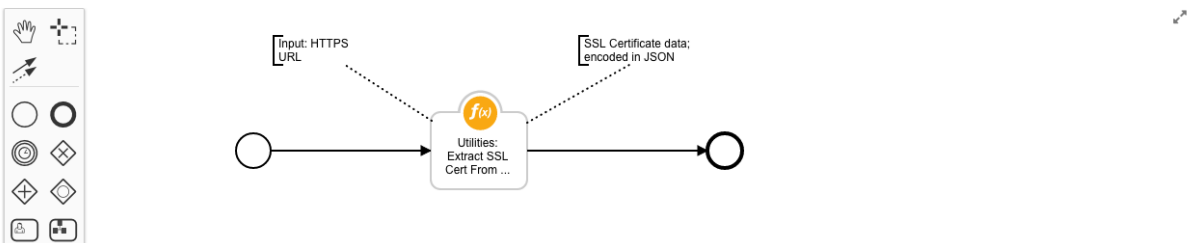
Name * Example: Extract SSL Cert from URL

API Name * example_extract_ssl_cert_from_url

Description This workflow takes in a HTTPS URL and attempts to acquire its Certificate, saving it as an artifact. The workflow runs at the artifact level

Object Type * Artifact

Creator Orchestration Engine
Last Modified 07/17/2019 14:20
Last Modified By Orchestration Engine
Associated Rules Example: Extract SSL Certificate



- Inputs:
- Outputs:
- Example Pre-Process Script:
- Example Post-Process Script:

Function - Utilities: Get Contact Info

Retrieves contact information of the owner and members of an incident or task.

Resilient Dashboards Simulations Incidents **Create** Admin User Test Organization

Customization Settings

Layouts Rules Scripts **Workflows** Functions Message Destinations Phases & Tasks Incident Types Breach Artifacts

Workflows / Example: Get Incident Contact Info Cancel Save & Close Save

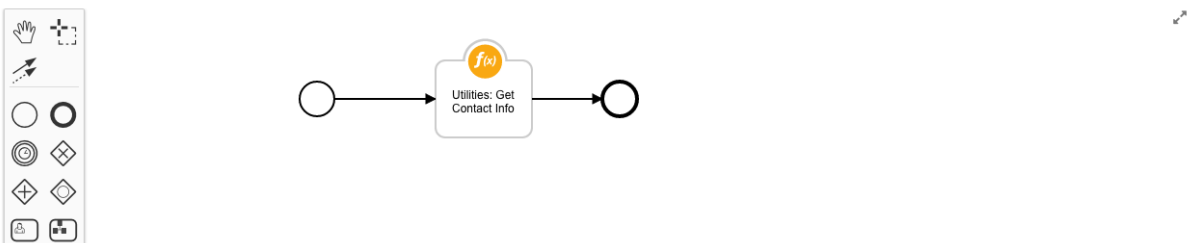
Name * Example: Get Incident Contact Info

API Name * example_get_incident_contact_info

Description Get owner and member contact information for an Incident

Object Type * Incident

Creator Orchestration Engine
Last Modified 07/17/2019 14:20
Last Modified By Orchestration Engine
Associated Rules Example: Get Incident Contact Info



- Inputs:
- Outputs:

- Example Pre-Process Script:
- Example Post-Process Script:

Function - Utilities: JSON2HTML

Produces an HTML representation of JSON data. All data is converted into tables of key / value pairs or lists.

Provides an optional parameter `json2html_keys` to limit the JSON data to display.

For the example below, specifying `key1.key2.key3` only converts the JSON data associated with that key path.

- Inputs:
- Outputs:
- Example Pre-Process Script:
- Example Post-Process Script:

Function - Utilities: Parse SSL Certificate

This function produces the structured data from a provided SSL certificate. Three inputs are accepted by the function. There are two defined ways to use this function for parsing certificates.

Option 1 involves providing a JSON-encoded representation of a certificate. In this case the certificate input parameter should be this JSON string.

Option 2 involves providing a certificate file for parsing. When the rule is triggered on an artifact, both the incident_id for that incident and the artifact_id for the specified certificate file must be provided.

NOTE: The Parse SSL Certificate function expects a certificate of type PEM. If you require a way to get a PEM formatted certificate from a URL, consider using this in conjunction with the Extract SSL Cert from URL function.

Customization Settings

[Layouts](#)
[Rules](#)
[Scripts](#)
[Workflows](#)
[Functions](#)
[Message Destinations](#)
[Phases & Tasks](#)
[Incident Types](#)
[Breach](#)
[Artifacts](#)

Workflows / Example: Parse SSL Certificate

[Cancel](#)
[Save & Close](#)
[Save](#)

Name *

Example: Parse SSL Certificate

API Name *

example_parse_ssl_certificate

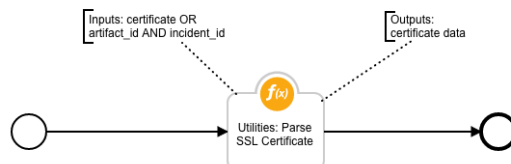
Description

An example workflow that takes in a PEM encoded SSL certificate as input and returns structured information about the certificate.

Object Type *

Artifact

Creator Orchestration Engine
Last Modified 07/17/2019 14:20
Last Modified By Orchestration Engine
Associated Rules [Example: Parse SSL Certificate](#)



- Inputs:
- Outputs:
- Example Pre-Process Script:
- Example Post-Process Script:

Function - Utilities: PDFiD

Produces summary information about the structure of a PDF file, using Didier Stevens' pdfid (<https://blog.didierstevens.com/programs/pdf-tools/>). The PDF file content should be provided as a base64-encoded string, for example the output from the "Attachment to Base64" function.

This function is useful in initial triage of suspicious email attachments and other files. It allows you to identify PDF documents that contain (for example) JavaScript or that execute an action when opened. PDFiD also handles name obfuscation. The combination of PDF automatic action and JavaScript makes a document very suspicious.

Customization Settings

[Layouts](#)
[Rules](#)
[Scripts](#)
[Workflows](#)
[Functions](#)
[Message Destinations](#)
[Phases & Tasks](#)
[Incident Types](#)
[Breach](#)
[Artifacts](#)

Workflows / Example: PDFiD

[Cancel](#)
[Save & Close](#)
[Save](#)

Name *

Example: PDFiD

API Name *

example_pdfid

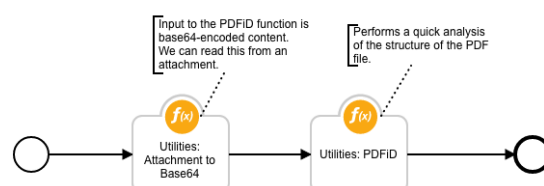
Description

An example of using the PDFiD function to get summary information about a PDF file attachment.

Object Type *

Attachment

Creator Orchestration Engine
Last Modified 07/17/2019 14:20
Last Modified By Orchestration Engine
Associated Rules [Example: PDFiD](#)



- Inputs:
- Outputs:
- Example Pre-Process Script:
- Example Post-Process Script:

Function - Utilities: Resilient Search

This function searches the Resilient platform for incident data according to the criteria specified, and returns the results to your workflow. It can be used to find incidents containing data that matches any string, incidents currently assigned to a given user, or a very wide range of other search conditions.

NOTE: The search results may include data from incidents that the current Resilient user (the person who triggered the workflow) cannot access. Often your Resilient users have the **Default** role that allows them to only see incidents where they are members. This function runs with the permissions of your app account, which typically may have much wider access privileges. **Use with caution, to avoid information disclosure.**

The screenshot shows the Resilient platform interface. At the top, there's a navigation bar with 'resilient' logo and tabs for Dashboards, Simulations, Incidents, and Create. Below this is a 'Customization Settings' section with tabs for Layouts, Rules, Scripts, Workflows, Functions, Message Destinations, Phases & Tasks, Incident Types, Breach, and Artifacts. The 'Workflows' tab is selected, showing a list of workflows with 'Example: Resilient Search' highlighted. To the right of the workflow list are buttons for 'Cancel', 'Save & Close', and 'Save'. Below the workflow list, there's a form for editing the workflow. The form includes fields for Name (Example: Resilient Search), API Name (example_resilient_search), Description (An example of searching Resilient.), and Object Type (Attachment). To the right of the form, there's a table with metadata: Creator (Orchestration Engine), Last Modified (07/17/2019 14:20), Last Modified By (Orchestration Engine), and Associated Rules (Example: Resilient Search). Below the form, there's a diagram showing a workflow with a start node, a function node labeled 'Utilities: Resilient Search', and an end node. A callout box points to the function node with the text: 'Search for any Resilient data. This example searches for all incidents that have an attachment with the same name.'

- Inputs:
- Outputs:
- Example Pre-Process Script:
- Example Post-Process Script:

Function - Utilities: Shell Command

This function allows your workflows to execute shell-scripts locally or remotely, and return the result into the workflow. The results include the **stdout** and **stderr** streams, the return code, and information about the execution time. If the output of the shell script is JSON, it is returned as structured data. Results can then be added to the incident as file attachments, artifacts, data tables, or any other uses.

These shell commands can be run on any linux or windows platform. Different modes supported:

- Remote Linux execution
- Remote Windows command and powershell execution
- Local command execution of Linux commands such as nslookup, dig, traceroute and whois

- Local execution of Windows Powershell commands if resilient-circuits is installed on a Windows platform.

Resilient Dashboards Simulations Incidents **Create** Admin User Test Organization

Customization Settings

Layouts Rules Scripts **Workflows** Functions Message Destinations Phases & Tasks Incident Types Breach Artifacts

Workflows / Example: Shell Command Cancel Save & Close Save

Name * Example: Shell Command

API Name * example_shell_command

Description An example running shell commands for integration

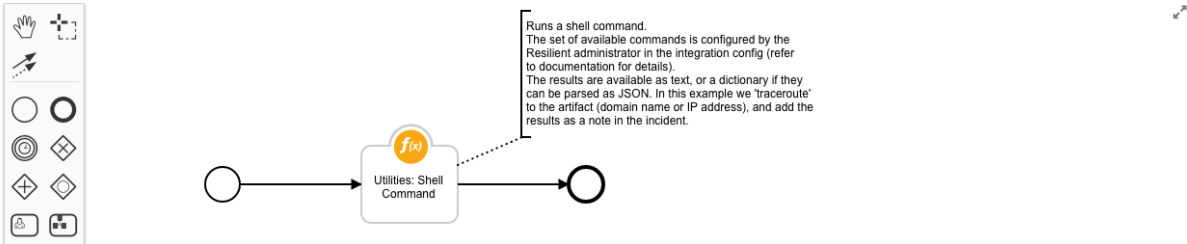
Object Type * Artifact

Creator Orchestration Engine

Last Modified 07/17/2019 14:20

Last Modified By Orchestration Engine

Associated Rules Example: Shell Command



- For security, the list of available shell commands must be **configured explicitly by the administrator**. To do this, edit the [fn_utilities] section of the **app.config** file.
- **NOTE:** The parameter values , , may contain spaces, dashes and other characters. In your command configuration, they must be surrounded with double-quotes. Failure to properly quote your command parameters creates a security risk, since the parameter values usually come from artifacts and other untrusted data.

For local and remote Windows environments:

- Remote commands must specify a target Windows machine that has Windows Remote Management (WinRM) enabled. This can be done by running **winrm qc** in the remote computer's command prompt.
- Remote shells have a max memory that may not be sufficient to run your script; to change this value you must set **MaxMemoryPerShellMB**.
- For remote powershell scripts, the **shell_param1**, **shell_param2** and **shell_param3** values map to **\$args[0]**, **\$args[1]**, and **\$args[2]** respectively in the Powershell script.

app.config examples:

- Linux Operating Systems basic examples:

```
# Remote Linux and Windows servers:
remote_computer=(usr1:password@192.168.1.186)
remote_computer_windows=(usr2:password@192.168.1.184)

# Remote Windows commands:
tracert_windows_ps=[\Users\ms\tracert.ps1]
tracert_windows_cmd=[tracert.exe -h 10 ]

# Remote Linux command:
tracert=(tracert -m 10 ")

# Local Linux server commands:
```

```
nslookup=nslookup ""
dig=dig ""
traceroute=traceroute -m 15 ""
```

- The following examples use the Volatility forensics framework. The first parameter is filename of the memory image, assuming \$VOLATILITY_LOCATION is set in the environment (such as in the system unit configuration). The second parameter is the Volatility profile ("Win7SP0x64" etc).

```
imageinfo=python /path/to/vol.py -f "" imageinfo -- output=json
kdbgscan=python /path/to/vol.py -f "" -- profile="" kdbgscan --output=json
psscan=python /path/to/vol.py -f "" -- profile="" psscan --output=json
dlllist=python /path/to/vol.py -f "" -- profile="" dlllist --output=json
```

Running Powershell Scripts Remotely:

To configure running scripts remotely, the user must make these changes to the config file:

- Specify acceptable powershell compatible extensions of script files, comma separated:
 - `remote_powershell_extensions=ps1,psc1`
- Specify the transport authentication method:
 - `remote_auth_transport=ntlm`
- Specify remote commands in the config file wrapped in square brackets []:
 - `remote_command=[C:\remote_directory\remote_script.ps1]`
- Specify a remote computer in the config file to run the script wrapped in parentheses ():
 - `remote_computer=(username:password@server)`

Examples of remote commands:

```
# Remote commands
remote_command1=[C:\scripts\remote_script.ps1]
remote_command2=[C:\scripts\another_script.ps1]

# Remote computers
remote_computer1=(domain\administrator:password@server1)
remote_computer2=(domain\admin:P@ssw0rd@server2)
```

- These remote commands can then be run in the workflow using the syntax `remote_command:remote_computer` as the input for `shell_command`. Examples:
 - `remote_command1:remote_computer1` runs `remote_command1` remotely on `remote_computer1`
 - `remote_command2:remote_computer1` runs `remote_command2` remotely on `remote_computer1`

- ▶ Inputs:
- ▶ Outputs:
- ▶ Example Pre-Process Script:
- ▶ Example Post-Process Script:

Function - Utilities: String to Attachment

Creates a new file (.txt) attachment in the incident or task from a string that your workflow provides as input.

The screenshot shows the 'Customization Settings' page in the Resilient interface. The 'Workflows' tab is selected. The workflow 'Example: String to Attachment' is being edited. The form includes fields for Name, API Name, Description, and Object Type. The Name is 'Example: String to Attachment', API Name is 'example_string_to_attachment', Description is 'An example of creating an attachment from an inputted string', and Object Type is 'Artifact'. On the right, metadata shows the Creator as 'Orchestration Engine', Last Modified as '07/17/2019 14:20', Last Modified By as 'Orchestration Engine', and Associated Rules as 'Example: String to Attachment'. Below the form is a visual workflow diagram showing a start node, a function node labeled 'Utilities: String to Attachment', and an end node. A toolbar on the left contains various icons for workflow construction.

- Inputs:
- Outputs:
- Example Pre-Process Script:
- Example Post-Process Script:

Function - Utilities: Timer

This function implements a timer (sleep) function that when called from a workflow causes the workflow to pause for the specified amount of time. The function takes one of two parameters as input: **utilities_time** or **utilities_epoch**.

The **utilities_time** parameter is a string that specifies the total amount of time to pause. The input string is of format **time value** concatenated with a **time unit** character, where character is:

- **s** for seconds
- **m** for minutes
- **h** for hours
- **d** for days

For example: **30s** = 30 seconds; **20m** = 20 minutes; **5h** = 5 hours; **6d** = 6 days

The **utilities_epoch** parameter is the epoch time which the timer function should sleep until that time has passed. An epoch time value is returned from the date time picker UI widget.

The timer function breaks down the total amount of time to pause into smaller sleep time intervals and checks in-between these sleep intervals whether the workflow has been terminated while the function is running. If the workflow has been terminated, the function will end execution.

Customization Settings

Layouts Rules Scripts **Workflows** Functions Message Destinations Phases & Tasks Incident Types Breach Artifacts

Workflows / Example: Timer

Cancel Save & Close Save

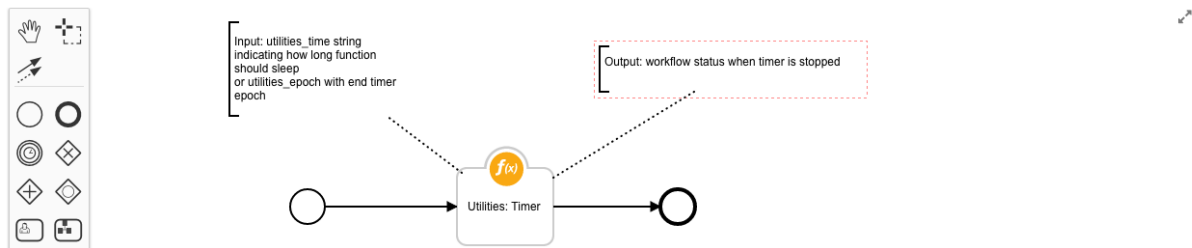
Name * Example: Timer

API Name * example_timer

Description This example workflow demonstrates how to call the Utilities Timer function using an epoch time as input to define when the timer should end.

Object Type * Incident

Creator Orchestration Engine
Last Modified 07/17/2019 15:44
Last Modified By Admin User
Associated Rules Example: Timer Epoch



- Inputs:
- Outputs:
- Example Pre-Process Script:
- Example Post-Process Script:

Function - Utilities: XML Transformation

Transforms an XML document using a pre-defined xsl stylesheet and returns the resulting content.

Customization Settings

Layouts Rules Scripts **Workflows** Functions Message Destinations Phases & Tasks Incident Types Breach Artifacts

Workflows / Example: XML Transformation

Cancel Save & Close Save

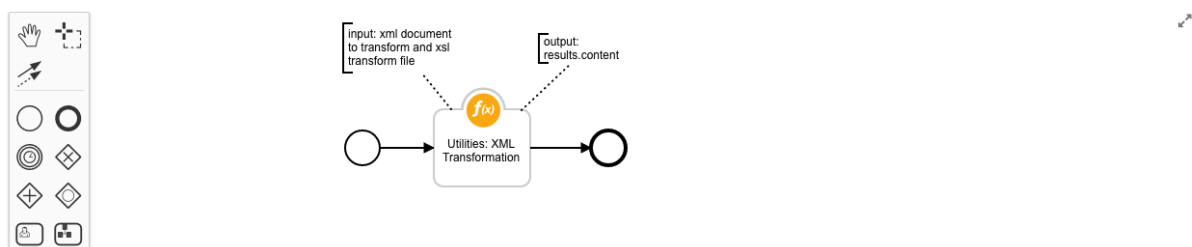
Name * Example: XML Transformation

API Name * example_xml_transformation

Description Transform an XML document using a defined xsl transform file

Object Type * Artifact

Creator Orchestration Engine
Last Modified 07/17/2019 14:20
Last Modified By Orchestration Engine
Associated Rules Example: XML Transformation



For App Host Environments:

- Set the app.config **xml_stylesheet_dir** setting as follows:

xml_stylesheet_dir= /var/rescircuits/xmltransformation
- Add your transformation file to the App Configuration tab to refer to the same directory as used in **xml_stylesheet_dir**.

h1>fn_utilities

Status: Ready For Use!

[Details](#)
[Customizations](#)
[Configuration](#)

h2>App Settings / cdcatalog.xslt

Edit the settings in the file below. File Path specifies a directory path starting at root. If changing location, the system creates the directory if it does not exist. Use a forward slash (/) only to place the file at the root directory. When done, click Save and Push Changes to implement your changes and restart the app.

Cancel

Save and Push Changes

Created Date: 2020-04-23 13:22
Last Modified Date: 2020-04-23 13:22

h3>File Name

cdcatalog.xslt

h3>File Path

/var/rescircuits/xmltransformation

h3>File Description

Purpose of the file.

[Show more](#)

h3>File Content

Text or code as appropriate.

Theme light

File Type Plain Text

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3   <xsl:template match="/">
4     <html>
5       <body>
6         <h2>My CD Collection</h2>
7         <table border="1">
8           <tr bgcolor="#9acd32">
9             <th style="text-align:left">Title</th>
10            <th style="text-align:left">Artist</th>
11          </tr>
12          <xsl:for-each select="catalog/cd">
13            <tr>
14              <td><xsl:value-of select="title"/></td>
15              <td><xsl:value-of select="artist"/></td>
16            </tr>
17          </xsl:for-each>
18        </table>
19      </body>
20    </html>
21  </template>
22 </xsl:stylesheet>
```

- ▶ Inputs:
- ▶ Outputs:
- ▶ Example Pre-Process Script:
- ▶ Example Post-Process Script:

h2>Rules

| Rule Name | Object | Workflow Triggered |
|--|------------|---------------------------------------|
| Example: (Artifact) Attachment to Base64 | artifact | example_artifact_attachment_to_base64 |
| Example: Attachment Hash | attachment | example_attachment_hash |
| Example: Attachment to Base64 | attachment | example_attachment_to_base64 |
| Example: Call REST API | artifact | example_call_rest_api |
| Example: Domain Distance | artifact | example_domain_distance |
| Example: Email Parsing (Artifact) | artifact | example_email_parsing_artifact |
| Example: Email Parsing (Attachment) | attachment | example_email_parsing_attachment |
| Example: Expand URL | artifact | utilities_expand_url |
| Example: Extract SSL Certificate | artifact | example_extract_ssl_cert_from_url |
| Example: Get Incident Contact Info | incident | example_get_incident_contact_info |
| Example: JSON2HTML | artifact | example_json2html |
| Example: Parse SSL Certificate | artifact | example_parse_ssl_certificate |

| Rule Name | Object | Workflow Triggered |
|-------------------------------|------------|--|
| Example: PDFiD | attachment | example_pdfid |
| Example: Resilient Search | attachment | example_resilient_search |
| Example: Shell Command | artifact | example_shell_command |
| Example: String to Attachment | artifact | example_string_to_attachment |
| Example: Timer Epoch | incident | example_timer |
| Example: Timers in Parallel | incident | example_timer_parallel |
| Example: Use Excel Data | attachment | example_create_artifacts_from_excel_data |
| Example: XML Transformation | artifact | example_xml_transformation |
| Example: Zip Extract | attachment | example_zip_to_artifact |
| Example: Zip List | attachment | example_zip_list |