IBM Resilient

Incident Response Platform

EMAIL CONNECTOR INSTALLATION AND CONFIGURATION GUIDE v2.2

Resilient Incident Response Platform Email Connector Installation and Configuration Guide

Version	Publication	Notes
2.2	December 2017	Documented product enhancements: RHEL based platform, EWS protocol, and discard utility.
2.1	July 2017	Added a note about editing property values directly in the configuration file.
2.1	July 2016	Initial release.

Table of Contents

1. Introduction

The Resilient Email Connector is a flexible component that enables organizations to easily integrate the Resilient platform with a variety of processes and systems that are generating security alerts. Individuals and systems can easily generate incidents in the Resilient platform simply by sending an email message.

Organizations have full control over how those inbound email messages are parsed and processed to set the values of fields on the resulting Resilient incidents. Attachments to the email can be included as attachments on the incident. Incident artifacts can be populated based on contents of the email message. The incident type can also be set, and particular individuals or groups can be assigned, based on the source or the contents of the email message. The result is the automatic generation of an appropriate runbook ready to go when the members receive their notification and log into the platform.

A common use case is for enterprises to configure their Security Incident and Event Management (SIEM) and Intrusion Detection Systems (IDS) to generate email alerts to a particular mailbox. The Resilient Email Connector then consumes those emails immediately generating appropriate incidents in the Resilient platform.

The Connector communicates with your email server using IMAP, EWS, and SMTP protocols, and communicates with your Resilient platform using the IRHub, which is the container for building connectors to the Resilient platform. The Resilient platform can be on premises or remote as in a SaaS configuration.

2. Before You Start

The installation procedure includes installation of both the IRHub and Email Connector, and sets the default email settings. You can then customize those settings as needed.

NOTE: If you have a previous version of the Email Connector, follow the instructions in Migration.

Before you start the installation, make sure you have the following:

- Host system with Red Hat Enterprise Linux 7.4 or later. You must have administrative privileges. If you have an on-premises installation, this can be the same system as the Resilient platform.
- Console access (e.g., SSH client such as PuTTY) to the host system.
- Email server that supports the IMAP or Exchange Web Services (EWS) protocol. The email server must be reachable from the host system by its hostname.
    - If using IMAP, you need to know the IMAP property values of hostname, username, password, port, and encryption settings (StartTLS, TLS, or plain).
    - If using EWS, you need to know the EWS property values of endpoint, username, and password. Exchange Server 2013 or later is also required.
    - If using SMTP, you need to know the SMTP property values of hostname, username, password, email address (if different from username), port, connection (StartTLS, TLS, or plain) and authentication (true or false).

    NOTE: The email server must support the SMTP protocol if you want to send confirmation emails. Alternatively, you can specify another SMTP server with a different mail domain to relay the confirmation email to your IMAP or EWS mail server.
- Access to the Resilient platform v27 or higher, whether on-premises or SaaS. You must be able to connect to the server by its hostname. You can check the Resilient platform version by logging in to the platform and clicking on the About section under your username.
- Resilient platform account that can create incidents. You must know the credentials of that account.

If you plan to parse the emails to customize incident reporting, you should identify the fields, along with type and values, required for creating incidents. See Customizing Incident Creation for more information.

If you plan to have confirmation emails, you need the addresses to send the success confirmation emails and error confirmation emails. These can be different addresses.

NOTE: Make sure you have completed the Email Connector checklist provided to you by Customer Success before installing the software. If needed, you can find the checklist at the Customer Success Hub. If you do not have an account, you can contact Customer Success at support@resilientsystems.com.

3. Installation

The installation procedure consists of installing IRHub and Email Connector then running scripts to configure your email server.

Perform the following:

1. Download the IRHub run file and Email Connector package from the Resilient repository to the system that is to host the Connector. You can contact support@resilientsystems.com to get the required credentials.
2. Establish a command shell on the host computer (e.g., use an SSH client such as PuTTY).
3. Install the IRHub run file by typing the following command, where <*version*> is the run file version:

   ```
   sudo bash irhub-<version>.run
   ```

4. When prompted, enter the name of the Resilient platform, user, and password to complete the configuration. Note that this user must be allowed to create incidents on the Resilient platform.
5. If prompted, click Yes to trust the certificates. Local connections, such as installing the IRHub on the on-premises Resilient platform, are always trusted. However, if you connect to a remote Resilient platform, such as app.resilientsystems.com, you must trust the certificate explicitly.
6. Select the Resilient account (in the form of an email address) to be used by IRHub.
7. If the user belongs to multiple organizations on the Resilient platform, you must select a default organization.
8. If you are installing the IRHub but not using a Resilient account (i.e., co3admin or resadmin), you are prompted to manually add that user to the irhubadmin group in order to connect to the IRHub console. To do this use the following command:

   ```
   sudo usermod -a -G irhubadmin USERNAME
   ```

9. Exit the terminal session and reestablish a connection, as described in step 2.
10. Install the Email Connector package using the following command, where <*version*> is the run file version.

    ```
    sudo rpm -i irhub-mail-<version>.rpm
    ```

11. If using the IMAP protocol, run the IMAP script to configure the email account to monitor by entering the following command and following the prompts.

    ```
    sudo irhub-imap-cfg
    ```

    As prompted, enter the following information:

    - IMAP mail server host name; for example, mail.example.com
    - Trust the certificate (only prompted if the certificate is untrusted)
    - IMAP username; for example, resilient@example.com
    - IMAP user password

    The script concludes by stating the location of the configuration file. For example:

    ```
    Selecting mailbox INBOX OK

    IMAP configuration settings were written to
    /usr/share/irhub/etc/irhub.mail.cfg
    ```

12. If using the EWS protocol, run the EWS script to configure the email account to monitor by entering the following command and following the prompts.

    ```
    sudo irhub-ews-cfg
    ```

    As prompted, enter the following information:

    - EWS endpoint; for example, https://mail.example.com/ews/exchange.asmx
    - Trust the certificate (only prompted if the certificate is untrusted)
    - EWS username; for example, resilient@example.com
      NOTE: it must be in email format. Domain\Username format does not work.
    - EWS user password

    The EWS script automatically sets the mail_protocol property to EWS. If using the EWS script to make changes after the initial installation, make sure to restart the IRHub for the updates to take effect.

    The script concludes by stating the location of the configuration file. For example:

    ```
    Using the following settings:

    Endpoint = https://mail.example.com/ews/exchange.asmx, Mailbox =
    Inbox

    EWS configuration settings were written to
    /usr/share/irhub/etc/irhub.mail.cfg
    ```

13. If you wish to configure the email account to send confirmation email, run the SMTP script.

    ```
    sudo irhub-smtp-cfg
    ```

    NOTE: If you choose to not set up an SMTP account, you should set reply_sendconfirmation to false, as described in Customizing Email Connector Properties; otherwise, the log will contain entries that reply addresses were not found.

    As prompted, enter the following information:

    - SMTP mail server host name; for example, mail.example.com
    - SMTP username; for example, IRTeam@example.com
    - SMTP password
    - A “From” email address. Typically, this is the same as SMTP username (e.g., IRTeam@example.com). This field is required to send confirmation emails.

    The script concludes by stating the location of the file containing the configuration settings. For example:

    ```
    SMTP configuration settings were written to
    /usr/share/irhub/etc/irhub.mail.cfg
    ```

14. Restart IRHub as follows:

    ```
    sudo systemctl restart irhub
    ```

This concludes the installation.

This procedure configures the basic settings and uses default values for the advanced properties. If you need to configure the advanced properties, see Configuration. You can also run the EWS, IMAP, or SMTP scripts at a later time to change the settings.

At this time, users can send email to the IMAP or EWS email address that you specified. By default, the Connector uses the subject of the email as the incident name, and the body of the email as the description. If this is satisfactory, you do not need to perform any of the additional configuration procedures listed in this guide. Instead, you can inform your users on the new email account where they need to send incident reports.

If you wish to configure advanced Email Connector properties, see Customizing Email Connector Properties.

Optionally, you can customize how the email is used to create an incident, as described in Customizing Incident Creation.

4. Configuration

You can run the IMAP, EWS, and SMTP scripts that were used in the installation procedure to reconfigure the email account to monitor and send confirmation emails. As described in the Installation section, establish a command shell on the host computer using an SSH client, log into IRHub then run each script using the sudo *script-name* command. To implement your changes, you must restart the IRHub when done.

To configure the advanced email properties, use the Apache Karaf commands, which are described in Advanced Configuration Commands then use the table in Customizing Email Connector Properties to set each property appropriately. In addition, you can customize how the email is used to create an incident, as described in Customizing Incident Creation.

4.1. Advanced Configuration Commands

You can set the properties specific to your email server and Resilient platform using the Apache Karaf commands from within the IRHub console. This section provides instructions on setting and listing the properties.

Refer to the Apache Karaf guide for list of all the commands you can use to get more information about the IRHub and installed Connector. The following is a list of the basic commands you need to configure the Connector:

- To log into the IRHub, establish a command shell on the host computer (e.g., use an SSH client such as PuTTY) and type **irhub**.
- To confirm that the Connector is installed:

  ```
  irhub> list | grep 'IRHub Mail Connector'
  ```

- To view all the properties that are required by the Connector:

  ```
  irhub> property-list -p irhub.mail |sort
  ```

  NOTE: The pipe (|) operator can be used to combine commands in IRHub, making it easy to view groups of properties. For example, to list all IMAP related properties alphabetically, enter the following command:

  ```
  irhub> property-list -p irhub.mail |grep imap |sort
  ```

  

- Use the property-set command to set the properties individually. For example, the following command configures the IMAP server to prevent reply confirmation messages:

  ```
  irhub> property-set -p irhub.mail reply_sendconfirmation false
  ```

- Start the Connector for the changes to take effect.

  ```
  irhub> restart 'IRHub Mail Connector'
  ```

4.2. Customizing Email Connector Properties

You can edit the advanced properties using the following procedure.

If you choose to edit the property values directly in the config file (/usr/share/irhub/etc/irhub.mail.cfg), make sure to add a backslash (\) to escape another backslash. For example, use the following syntax to configure the imap_password property as **\Passw0rd**:

```
imap_password=\\Passw0rd
```

1. Establish a command shell on the host computer (e.g., use an SSH client such as PuTTY) and type **irhub** to log in.
2. Refer to the table in this section to determine which properties need to be reconfigured.
3. Use the property-set command to set the properties individually. For example, the following command configures the IMAP server to imap.example.com:
   ```
   irhub> property-set -p irhub.mail imap_server imap.example.com
   ```
4. When done, restart IRHub as follows:
   ```
   irhub> restart 'IRHub Mail Connector'
   ```

Property	Description
client_org	Name of a Resilient organization different from the one selected during IRHub installation for incident creation. By default, this is **blank**, and applies only if the IRHub user belongs to multiple organizations. This provides additional flexibility to users so that they can post incidents to another organization without having to reconfigure the IRHub. The user must have permissions to create incidents in that organization.
ews_endpoint	URL of the EWS service, which is accessed by the Connector, for example: https://<hostname>/ews/exchange.asmx.
ews_folder	Name of the EWS mailbox (Inbox by default).
ews_password	EWS account password, used to access the mailbox.
ews_username	EWS account username, used to access the mailbox. This must be in email format; such as username@example.com.
felix.fileinstall.filename	Name of the configuration file that stores the properties. By default, this is /usr/share/irhub/etc/irhub.mail.cfg. This file cannot be changed.
imap_auth_plain_disable	By default (set to true), prevents use of the AUTHENTICATE PLAIN command.
imap_auth_ntlm_disable	By default (set to true), prevents use of the AUTHENTICATE NTLM command.
imap_enablessl	Encryption setting on the IMAP Server using TLS. This is set during installation.

Property	Description
imap_filter	Filter criteria to process for incident creation. By default, this is set to true, which means that all incoming emails are considered for incident creation. Optionally, administrators may set a filter such as *subject matches '.*(?i)incident.*'* which means that an email is considered for incident creation if it contains the word, *incident*, in the email subject. "?i" indicates case insensitivity. Refer to the mail-filter-expression guidelines for the Java Spring Framework to create your own filters.
imap_folder	Folder name used for email retrieval. By default, this is INBOX.
imap_password	Password of the IMAP user.
imap_port	IMAP port. Typically 143 for non-encrypted or 993 for secure connections.
imap_server	Name of the IMAP server used for email retrieval (e.g., imap.example.com).
imap_starttls	Option for connecting to the IMAP Server using STARTTLS. By default, this is set to true.
imap_username	Username of the account used to retrieve emails for incident creation (e.g., user@example.com).
incident_post_attempts	Number of times the system attempts to post to the Resilient platform. The default is 3.
incident_post_timeout	Optional. Interval period in milliseconds to wait between each posting attempt. The default is 15000.
incident_template	Name of the optional JSON file that can be used to add more incident details (e.g., category). A sample file (IRMailIncidentTemplate.json) is available in /usr/share/irhub/etc. The path must be specified if using a non-default folder (full path, or relative to usr/share/irhub). Refer to Customizing Incident Creation for more details.
keystore_file	Name of the Java keystore that stores the certificates of the IMAP and SMTP servers. The default is etc/irhub.trustsore, which is created during installation. The irhub-imap-cfg and irhub-smtp-cfg utilities write to this store during configuration. To set it manually, refer to How to Add a Certificate to the Keystore.
keystore_password	Password for the keystore.
mail_allowattachments	Flag indicating whether attachments can be included in the email messages for incident creation. By default, this is true.
mail_debug	System setting to enable debugging to help troubleshooting any mail server connection related problems. By default, this is set to false.
mail_protocol	Determines the email protocol to use. Valid values are IMAP (default) or EWS.

Property	Description
mail_script	Name of the XML file, relative to the IRHub installation folder. It contains the script to process incoming email messages, transforming them in a format that can be used by the Resilient API. The default file, MailParserDefault.xml, is installed in the /usr/share/irhub/etc folder. It creates Resilient incidents using: • Email Subject as the Incident Name • Email Body text as the Incident Description • Email Sender as the Incident Reporter Refer to Customizing Incident Creation on updating these scripts. **NOTE**: If you wish to customize the parser script, we recommend that you create a new file (e.g., MailParserCustom.xml) based on the default file, and specify it in the Email Connector properties (see Customizing Email Connector Properties).
Note about the reply settings	With each reply setting listed in this table, you can specify a hardcoded email address (e.g., admin@example.com) or an expression (e.g., payload.reporter). If you prefer to use an expression such as payload.reporter, you can set it as follows: `property-set -p irhub.mail reply_bcc spel:payload.reporter` At this point, **payload** refers to the incident created; thus, all existing incident fields can be accessed. For more information about expressions, refer to Spring Expression Language.
reply_bcc	Email address of the user BCC'd on the confirmation email.
reply_cc	Email address of the user CC'd on the confirmation email.
reply_errorcc	Email address of the recipient of any error messages generated by the system.
reply_from	FROM email address that is displayed in the confirmation email. The smtp_username must be able to send emails with this address.
reply_replyto	Email address of the user to Reply-to for the confirmation email; for example, **donotreply@example.com**. You should modify it for your organization.
reply_sendconfirmation	Indicates whether to send confirmation emails. By default, this is false. You can enable it after successfully running the irhub-smtp-config utility.
reply_to	Email address of the confirmation email recipient. By default, this is sent to incident reporter (*payload.reporter*).
service.pid	Name of the Resilient service on the system. This is irhub.mail by default, and must not be changed.

Property	Description
smtp_auth	Setting on the SMTP Server that determines whether user authentication is required to send emails. This is set during installation.
smtp_enablessl	Encryption setting on the SMTP Server using TLS. This is set during installation.
smtp_password	Password of the SMTP user.
smtp_port	SMTP port. Typically, this is 25 for non-encrypted or 587 for secure connections.
smtp_server	Name of the SMTP server (e.g., smtp.example.com) used to send confirmation emails to users. On-premises customers may use the same server and credentials that they have configured with their platform.
smtp_starttls	Encryption setting on the SMTP Server using STARTTLS. This is set during installation.
smtp_username	Account on the SMTP Server used for authentication. For example, smtpuser@example.com.
wlhosts	Hostnames of the IMAP, EWS, and SMTP servers that are whitelisted (explicitly trusted) if there is a Common Name (CN) mismatch between the hostname and its certificate. Typically, this is set while running the irhub-imap-cfg, irhub-ews-cfg, and irhub-smtp-cfg scripts. To set it manually, log in to the IRHub console and enclose the hostname within \Q and \E. For example: `property-set -p irhub.mail wlhosts \\Qsmtp2.example.com\\E`

4.3. Customizing Incident Creation

By default, the Connector uses the subject of the email as the Resilient incident name, and the body of the email as the description. Any attachments to the email are included to the incident. You can customize this behavior programmatically by parsing email content using JavaScript.

There are two files used to create incidents from emails:

- The mail_script property, as described in Customizing Email Connector Properties, references a mail parser XML file. The default file provides code to create incidents using the mail subject as the name and mail body text as the description. You can execute code to determine if the email should be used to create a new incident or update an existing incident, as well as parse more complex emails such as those generated by SIEMs, to create incidents. By default, the file is MailParserDefault.xml located in the usr/share/irhub/etc folder.

- The incident_template property, as described in Customizing Email Connector Properties, references a file called IRMailIncidentTemplate.json also in the usr/share/irhub/etc folder. This file is a JSON representation of a FullIncidentDataDTO and contains a default name and default description, as well as a default set of regulators. The fields in the template are applied first, and then the mail script overwrites them with the fields in the scripting section of the mail parser XML file. You should set in the template those fields that you expect to be constant or required when creating a new incident. Therefore, if a field is not defined in the XML file, the value defined in the template file is used when creating the incident.

This section describes how to parse the email using the map parsing XML file. The example scripts provided in the /usr/share/doc/irhub-mail/examples folder show how more complex emails generated by SIEMs can be parsed and used to create incidents. After you parse the mail script, you can test it using the irhub-script utility as described in Testing the Mail Script.

4.3.1. JavaScript in the XML Scripting Context

In order to provide scripting capabilities, the Connector implements Rhino, which is an open-source implementation of JavaScript. Refer to the Rhino documentation for information on how to use scripting in Rhino.

When inserting JavaScript code into the XML elements, it is recommended to put the code inside a <![CDATA[ ]]> tag. This prevents the data from within this tag from being interpreted as XML markup, which is not desired.

The mail parser XML file has the following structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<MailParser xmlns="http://www.resilientsystems.com/MailParser">
    <Query></Query>
    <Script></Script>
    <PostScript></PostScript>
</MailParser>
```

4.3.2. Data Types

The SimpleEmailDTO data type is in all the XML elements. FullIncidentDataDTO and SimpleClient are in the Script and PostScript elements only.

SimpleEmailDTO

To access the email content, a simplified version of the email is passed into the <Query>, <Script> and <PostScript> elements' scripting context. It is of type SimpleEmailDTO and can be accessed using the variable name, **mail**. It contains the following fields.

Variable	Type	Description
mail.from	List	Email sender.
mail.to	List	Email recipients.
mail.cc	List	CC'd email recipients.
mail.dateTime	String	Date and time of the email message in the yyyy-MM-dd HH:mm:ss format.
mail.subject	String	Subject of the email.
mail.bodyText	String	Text of the email body in plain text. If the email body is in HTML format, the plain text version of the email is used.
mail.attachments	List	List of attachment information. Contains an entry for each attachment of the email with the following information: **fileName**: Type is String. Name of the attachment. **contentType**: Type is String. Mime content type of the attachment. **post**: Type is Boolean. Whether the attachment should be posted to the incident. Default is true. **size**: Type is Number. File size, which may not be an exact measure of the content as it may not account for any transfer encoding of the content.
mail.headers	List	List that contains all headers from the original email as key/value pairs. For example, you can access the email message ID as follows: `mail.headers["Message-ID"];`

FullIncidentDataDTO

A FullIncidentDataDTO is passed into the <Script> elements' scripting context and can be accessed using the variable name, **incident**. The content depends on if the previous query successfully returned an existing incident:

- When a query successfully returns an incident, it is passed into the scripting context. All changes to it update the existing incident after processing of the <Script> element code.
- When a query does not return an incident or no query was executed, a new incident is created. The incident is loaded from the file, IRMailIncidentTemplate.json. To not execute a query, leave the <Query> elements contents empty or set to **null;**.

- When a query returns more than one incident, it means that a distinct incident does not exist and processing is aborted. Processing is also aborted if the script for the query is malformed and could not execute.

For additional information about the incident data type, refer to the FullIncidentDataDTO section in the Resilient REST API documentation.

SimpleClient

A REST client can be accessed by using the variable name **client**. This client is an instance of the Resilient SimpleClient class. To see the capabilities of SimpleClient, review the Java examples in the Resilient GitHub repository. These examples use the SimpleClient.java class for performing HTTP requests to the Resilient web application.

The class supports various methods for performing various requests, such as GET, PUT, and POST. Review the Resilient REST API documentation to see the available REST endpoint resources and associated data types.

The following example adds a row to a data table within the current incident, which may be useful in the <PostScript> element:

```
client.post("incidents/" + incident.id + "/table_data/" + {table_id} +
"/row_data", rowData);
```

NOTE: You should not perform any incident updates, such as PUT requests, within the <Script> element context. This can cause version differences and the Resilient platform to reject any updates later on.

4.3.3. Elements

There are three elements within the XML file, <Query>, <Script> and <PostScript>. Each can contain Rhino valid JavaScript code.

<Query> Element

The <Query> element is executed first. If the code in the <Query> element is empty or null, the Connector does not execute the query. By default, the <Query> element is set to null.

The result of the query is expected to be a list of conditions, which can contain multiple conditions to query for an incident. For details, refer to the ConditionsDTO section in the Resilient REST API documentation.

The following shows an example of the <Query> element:

```
<Query>
    [{
       method: "contains",
       field_name: "name",
       value: mail.subject
    }];
</Query>
```

<Script> Element

In the <Script> element scripting context, both “mail” and “incident” data are available and you can access their fields as discussed previously. Furthermore, you can use the “client” object to perform HTTP requests to the Resilient REST API. The standard script shipped with the Connector handles creating new incidents only. It maps the email subject as the incident name, the email body text as the incident description, and the email sender as the incident reporter.

The following shows an example of the <Script> element:

```
<![CDATA[
if(incident.id == null){
          incident.name = mail.subject;
          incident.description = mail.bodyText;
          incident.reporter = mail.from.toString().slice(1,-1);
          }
]]>
```

You can determine if the incident exists or is newly created by checking the incident.id field. Only existing incidents have an id. If the field is null, it is a newly created incident, which was loaded based off the IRMailIncidentTemplate.json file.

You can set and modify the incident fields as long as it complies with a FullIncidentDataDTO. The requirements could be different on each system, depending on whether there are any custom fields or if any required fields have to be set.

NOTE: If adding new artifacts, notes or attachments to the incident, the confirmation email to the creator includes the number of successfully created artifacts and notes as well as attachments.

You can use the utils.discardIncident() utility to prevent an incident from being created by the current email. For example:

```
<Script>
   <![CDATA[
   if(mail.subject == "test email"){
      incident.name = mail.subject;
      incident.description = mail.bodyText;
      incident.reporter = mail.from.toString().slice(1,-1);
   }
   else {
      utils.discardIncident();
   }
   ]]>
</Script>
```

<PostScript> Element

The <PostScript> element allows you to perform post-incident processing actions. It provides access to the newly generated or updated **incident**, **mail** property, and the REST **client** object. Therefore, you could post to the incident data that was not available when you ran the <Script> element. For example, you could add data table information, as shown in the SimpleClient example in Data Types.

There is no additional processing within the Connector after executing the PostScript element. This means changing incident data at this point does not have any effect unless the incident is updated using the REST client within the <PostScript> context.

You can find an example of how to add rows to a data table leveraging the PostScript element in the /usr/share/doc/irhub-mail/examples/scripts/PostScriptDataTables.xml file.

You can use the utils.incidentDiscarded method to verify that the discardIncident() utility discarded an email and prevented an incident from being created. For example:

```
<PostScript>
       <![CDATA[


           if(!utils.incidentDiscarded()) {
               // .....
               var table = getTableTypeDef(dataTableApiName);
               var table_id = table.get("id");
               var row = generate_row(table);
               client.post("incidents/" + incident.id + "/table_data/" +
table_id + "/row_data", row);
           }
    ]]>
</PostScript>
```

4.3.4. Testing the Mail Script

You can test your code using the provided irhub-script utility. The utility tests the code in your mail script. It does not identify data type mismatches between json and incident, and it does not verify that all required fields are set properly for your environment.

Once the mail script is tested, you should send an email to create an incident and verify that all fields in the incident are set as expected.

To execute the irhub-script utility, execute the following command from the IRHub host terminal window. The email.eml in the command is the representative email file that you wish to parse.

```
sudo irhub-script -email email.eml -script_file MailParserCustom.xml
```

The following is an example of using the utility.

```
sudo irhub-script -mime_file /usr/share/doc/irhub-mail/examples/test-
emails/SplunkAlert1.eml -script_file /usr/share/doc/irhub-
mail/examples/scripts/SplunkScript.xml
```

If your script interacts with the Resilient API using the REST client, you can add the required client settings as additional arguments to the utility.

```
sudo irhub-script -mime_file /usr/share/doc/irhub-mail/examples/test-
emails/SplunkAlert2.eml -script_file /usr/share/doc/irhub-
mail/examples/scripts/SplunkScript.xml –client_cfg_file
/usr/share/irhub/etc/irhub.cfg –truststore_file
/usr/share/irhub/etc/irhub.truststore –truststore_password changeit
```

If you do not set arguments, the script checks the IRHub default directory for a client configuration file (client-cfg_file) called /usr/share/irhub/etc/irhub.cfg as well as the default keystore in /usr/share/irhub/etc/irhub.truststore in combination with default truststore password, **changeit**.

The following key-value pairs for accessing the Resilient Application via the REST interface are expected within the configuration file:

- email = user@example.com
- password = yourpasswd
- url = https://resilient.example.com
- defaultOrg = yourOrg

5. Uninstallation

To uninstall, you need to remove the Connector package from IRHub and then uninstall the IRHub as follows:

1. Remove the Connector from IRHub:

   ```
   sudo rpm –e irhub-mail
   ```

   The rpm -e keeps the configuration files in place so that they can be reused for future installs. If you wish to remove all the files, you must remove them manually.

2. Uninstall the IRHub:

   ```
   sudo rpm –e irhub
   ```

3. Uninstall the Resilient IBM JDK package:

   ```
   sudo rpm –e resilient-ibm-jdk-7
   ```

6. Migration

Previous releases of the Resilient Email Connector and IRHub were based on Debian Linux. Therefore, you cannot upgrade to the RHEL based release of the Email Connector and IRHub, but you can migrate your settings to a new installation.

Your Email Connector and IRHub must be at V2.1. If not, you need to upgrade to V2.1 before you can migrate.

Perform the following procedure to migrate your settings to V2.2:

1. Download the IRHub and Email Connector 2.2 migration package from the Resilient repository. You can contact support@resilientsystems.com to get the required credentials.

   The package is a zip file containing the migration scripts. It needs to be loaded onto both the Email Connector 2.1 and 2.2 host systems.

2. Unzip and upload the migration scripts to a folder in the V2.1 Email Connector host system.

3. Establish a command shell on the host computer (e.g., use an SSH client such as PuTTY).

4. Enter the following command to back up the settings.

   ```
   ./irhubBackup <backup_file_path>
   ```

5. Upload the migration scripts to a folder in the V2.2 Email Connector host system.

6. Establish a command shell on the host computer (e.g., use an SSH client such as PuTTY).

7. Enter the following command to import the settings. The *debian_backup_file_path* is the path to the IRHub backup file from the Debian IRHub server. The *redhat_backup_file_path* is a path you choose. The migrate command saves the current v2.2 IRHub/Email Connection configuration settings in this location before overwriting with the V2.1 configuration settings.

   ```
   ./irhubMigrate <debian_backup_file_path> <redhat_backup_file_path>
   ```

8. Restart the IRHub to complete the migration.

   ```
   sudo sytemctl restart irhub
   ```

You may wish to review your settings, as described in Customizing Email Connector Properties.

7. Troubleshooting

This section provides various procedures and commands that may be useful when troubleshooting issues.

7.1. How to Add a Certificate to the Keystore

The certificate of the email server must be trusted. This is done during the initial configuration by the irhub-imap-cfg and irhub-smtp-cfg scripts. We recommend that you run these scripts again should you need to trust the certificates. To manually trust a certificate, follow these steps:

1. Login to the IRHub console.
2. Note the values of the keystore_file and keystore_password properties. The following command displays both values:

   ```
   irhub> property-list -p irhub.mail |grep keystore
   ```

3. Obtain the certificate of the email server; for example, yourcert.crt.
4. Enter the following command in the terminal window on the same system as the Connector:

   ```
   sudo keytool -importcert -keystore <path_name_of_keystore_file> -file
   yourcert.crt -alias -yourcert
   ```

5. Enter the password, which is the value of the keystore_password property.
6. Restart the IRHub:

   ```
   sudo sytemctl restart irhub
   ```

7. Login to the IRHub and enter the log:display command.
8. Verify that you observe a "Certificate is trusted..." message in the logs.

7.2. How to Exit the IRHub Console

Frequently, you may need to log in to the IRHub console to view and change the properties. To exit the console, type logout from the IRHub prompt or use the CTRL+D shortcut on your keyboard.

NOTE: If using the log:tail command, you need to first use the CTRL+C shortcut to return to the IRHub prompt.

7.3. How to Change Log Levels

The pertinent log file for the Email Connector integration is **karaf.log** and it is located in the /usr/share/irhub/data/log folder. This file contains the output of the IRHub and the Connector.

By default, the log level is set to INFO. To view your current log level settings:

```
irhub>log:get
```

To change the log level, use the following command:

```
irhub>log:set DEBUG
```

The acceptable log levels are TRACE, DEBUG, INFO, WARN, ERROR, and FATAL.

7.4. Email Connector Disconnects from IMAP Server

In some cases, the Email Connector gets disconnected from the IMAP Server. This may happen if the imap_filter property is misconfigured or if there are a large number of unread emails in the mailbox. In such cases, we recommend that you reset the imap_filter property and use IMAP Server filters; e.g., filter messages to a particular mailbox folder and monitor that using the imap_folder property. You can reset the property as follows:

```
irhub> property-set -p irhub.mail imap_filter true

irhub> restart 'IRHub Mail Connector'
```

You can also filter the messages using the mail parser script as described in Customizing Incident Creation.

7.5. Useful Commands

Use the following command to check if you can send emails using SMTP. Once connected, you can issue standard SMTP commands to send emails.

```
openssl s_client -connect <SMTP_SERVER>:<SMTP_PORT> -starttls smtp
```

Use the following command to verify that you can check emails using IMAP. Once connected, you can issue standard IMAP commands to check emails.

```
openssl s_client -connect <IMAP_SERVER>:<IMAP_PORT> -starttls imap
```

If you need to change IRHub settings, use the following command:

```
sudo irhub-config
```

End of document