



Resilient Incident Response Platform

EMAIL OUTBOUND CONNECTOR GUIDE v2.1

© 2015 Resilient Systems, Inc. All rights reserved.

This guide and the software described in this guide are furnished under a license accompanying the software and may be used only in accordance with the terms of such license. By using this guide, you agree to the terms and conditions of that license.

Resilient and Resilient Systems are trademarks or registered trademarks of Resilient Systems, Inc. in the United States and other countries. All other product names mentioned herein may be trademarks or registered trademarks of their respective companies.

Published: December 2015

<https://www.resilientsystems.com>

Table of Contents

1.	Introduction	5
2.	Before You Start.....	5
3.	Installation	6
4.	Configuration	8
4.1.	Creating HTML Email Templates	8
4.1.1.	File Naming Convention	8
4.1.2.	Creating email body	8
4.1.3.	Adding Recipients, Headers and Subject	9
4.1.4.	Data Model	9
4.1.5.	Additional Methods	10
4.2.	Creating Actions	11
4.3.	Adding a Certificate to the Keystore.....	11
4.4.	Configuring Bi-Directional Email	12
4.5.	Advanced Configuration Commands	13
5.	Uninstallation	13
6.	Troubleshooting.....	14
6.1.	Managing the Log File.....	14
6.2.	How to Exit the IRHub Console.....	14
6.3.	Useful Commands.....	14

1. Introduction

The Email Outbound Connector processes actions triggered from the Resilient platform, converts them into email by using a template then sends the email using the SMTP protocol.

The Email Outbound Connector is an IRHub application and uses FreeMarker Template Engine (<http://freemarker.org/>) that allows for mapping Resilient action fields into preconfigured HTML templates. HTTP requests to the Resilient platform can also be done within the context of the template.

You can also use the Email Outbound Connector with the Resilient Email Connector to provide bi-directional email communication. For example, you can have recipients of an email sent from the Resilient platform via the Outbound Connector respond to that email, where the Email Connector identifies those responses and updates the respective incident.

2. Before You Start

The Email Outbound Connector communicates with your email server using SMTP, and with your Resilient platform using the IRHub. The Resilient platform can be on premises or remote as in a SaaS configuration.

The installation procedure includes installing the Email Outbound Connector to the IRHub, and setting the email settings.

Before you start the installation, check the following:

- IRHub V2.1 is installed and available. Refer to the *Resilient Incident Response Platform Email Connector Installation and Configuration Guide*, which contains the procedure to install and configure the IRHub.

NOTE: You need IRHub V2.1. Version 2.0 of IRHub is not compatible with this version (v2.1) of the Email Outbound Connector!

- You have administrative privileges to the IRHub host system and you have console access (e.g., SSH client such as PuTTY) to the host system.
- Your email server supports the SMTP protocol and is reachable from the host system by its hostname. You must also know the email server SMTP settings, including hostname, username, password, port, connection (StartTLS, TLS, or plain) and authentication (true or false).

NOTE: If the Email Connector is installed to the same IRHub, you can copy the SMTP settings from the Email Connector configuration file.

- The Resilient platform has the optional Action Module installed. For information on the module and actions in general, refer to the *Resilient Incident Response Platform Master Administrator Guide*. You may also need access as a master administrator to create actions.
- You should be familiar with HTML code and the FreeMarker template engine.

3. Installation

Perform the following to install the Email Outbound Connector to an IRHub instance:

1. Verify that IRHub V2.1 is installed and properly configured with a Resilient user account. For information, see the *Resilient Incident Response Platform Email Connector Installation and Configuration Guide v2.1*.
2. Optionally, log into the Resilient platform as a master administrator and create a message destination as described in the *Resilient Incident Response Platform Master Administrator Guide* with the following settings:
 - **Type:** Queue or Topic.
 - **Name:** Any name you choose. Be sure to note the resulting programmatic name, which you need to configure the Email Outbound Connector.
 - **Expect Acknowledgement:** Yes.
 - **Users:** If not done already, add the Resilient user account used by the IRHub.

NOTE: If you do not perform Step 2, the Email Outbound Connector automatically creates the queue on startup, based on the queue_name property, and adds the Resilient user from the IRHub as the queue user.

3. Locate the Email Outbound Connector package and download it to the system that hosts the IRHub.
4. Copy MailHtmlTemplate.ftl and smtpservice.irhub.cfg from the package to the IRHub etc folder, /usr/share/irhub/etc and change the file owner to group "irhub" and owner "irhub" as well as verify the file owner has at least read permissions.

```
$ sudo chown irhub:irhub /usr/share/irhub/etc/MailHtmlTemplate.ftl
$ sudo chown irhub:irhub /usr/share/irhub/etc/smtpservice.irhub.cfg
```

If you want to use the example template Mail.send_email.ftl, the same steps apply.

5. Using vi or another editor, edit smtpservice.irhub.cfg to add the following settings.

NOTE: The Email Connector might use the same SMTP settings. If installed on the IRHub, enter the values used by the Email Connector.

Property	Description
queue_name	If the queue name was created on the Resilient platform, set this field to the programmatic name of the message destination. If the queue does not exist on startup, it is created automatically, adding the IRHub Resilient user account as a member.
uuid_field_name	Not used unless you use the Email Outbound Connector with the Email Connector to leverage bi-directional email communication. In this case, you can use this field to store an ID that identifies emails and associated incidents. See Configuring Bi-Directional Email. If it is not blank and does not exist in the Resilient platform, this field is also created automatically on startup.

Property	Description
org_name	Name of a Resilient organization. By default, this is blank , and applies only if the IRHub Resilient 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.
html_template	Set to MailHtmlTemplate.ftl. DO NOT CHANGE.
html_path	Set to /etc. DO NOT CHANGE.
mail_debug	System setting to enable debugging that helps with troubleshooting any mail server connection related problems. By default, this is set to false.
smtp_server	Name of the SMTP server (e.g., smtp.example.com) used to send emails to users.
smtp_auth	Setting on the SMTP server that determines whether user authentication is required to send emails. Set to either true or false.
smtp_enablessl	Encryption setting on the SMTP Server using TLS. Set to either true or false.
smtp_port	SMTP port. Typically, this is 25 for non-encrypted or 587 for secure connections.
smtp_starttls	Encryption setting on the SMTP Server using STARTTLS. Set to either true or false.
smtp_from	Address used by the email account that sends the email, which appears in the From field.
smtp_username	Account on the SMTP Server used for authentication; for example, smtpuser@example.com, or domain\user\mailbox in Microsoft Exchange.
smtp_password	Password of the SMTP user.
keystore_file	Name of the Java keystore that stores the certificate of the SMTP server. The default is etc/irhub.truststore. To set it manually, refer to Adding a Certificate to the Keystore .
keystore_password	Password for the keystore. By default, this is changeit .
wlhosts	Supports wlhosts to trust certificates, other than mail server name. Also supports regex and wildcards. E.g. mailserver*\d*.example.com

- Copy irhub-smtp-service-kar-x.x.x.kar from the package to the IRHub deploy folder, /usr/share/irhub/deploy.
- Restart IRHub as follows:

```
$ sudo service irhub restart
```

- Log in to IRHub and verify that the status of the Outbound Email Actions Service is active. For example:

```
xxx | Active | 80 | x.x.x | Outbound EMail Actions Service
```

This concludes the installation.

4. Configuration

Once the Email Outbound Connector is installed, you need to create a least one HTML template. You can create multiple templates for different actions. Afterwards, the Resilient master administrator can create actions based on the templates.

4.1. Creating HTML Email Templates

The email templates define the content of the outbound email. You determine the static text and variables. You create email templates using the FreeMarker template engine.

The following sections describe the requirements for the template.

4.1.1. File Naming Convention

Name the template file `Mail.name.ftl`, where *name* is descriptive of the template. The *name* must be all lowercase. When creating an action, use *name* as the name of the action; however, the action name can be mixed case and any underscores are replaced by spaces. For example, a **Send Acknowledgement** action calls the **Mail.send_acknowledgement.ftl** template.

4.1.2. Creating email body

Any output within the template is considered as the email body. Therefore, you can create the email's HTML text by combining the template and data model and then printing it out in order to generate the output.

```
<!-- Create mail body text -->
<#assign body>
<table style="font-family: MyriadSet-Text; font-size: 14px; font-style:
normal;">
<tr><td><strong>Incident information for [ID:
${incident.id?c}]</strong></td></tr>
<tr><td><br/></td></tr>
<tr><td>${.now?date!"Date N/A"} ${.now?time!""}</td></tr>
<tr><td><br/><br/></td></tr>
<tr><td><strong>Overview</strong></td></tr>
<tr><td>Name:</td><td>${.data_model.incident.name!"-"}</td></tr>
<tr><td>Description:</td><td>${.data_model.incident.description!"-
"}</td></tr>
<tr><td><br/><br/></td></tr>
<tr><td>Additional
comments:</td><td>${.data_model.properties.mail_text!"-"}</td></tr>
<tr><td><br/><br/></td></tr>
<tr><td><small>Sent from Resilient Systems' Integration
Appliance</small></td></tr>
</table>
</#assign>

<!-- Print email output -->
```

```
${body}
```

4.1.3. Adding Recipients, Headers and Subject

You can use the `putToMail` method to add any key value pairs to the email header. It takes the email header name and value as arguments, and does not return any value. You need to use this method since email fields such as TO, CC, BCC and SUBJECT are also set within the email headers.

```
<!-- Add email headers -->
${putToMail("to", .data_model.properties.mail_to! "")}
${putToMail("cc", "staticrecipient@example.com")}
${putToMail("bcc", .data_model.properties.mail_bcc! "")}
${putToMail("subject", "Some email subject")}
```

4.1.4. Data Model

In the template context, you have access to the full `actionDataDTO` data model, which is described in the *Resilient Incident Response Platform Action Module Programmer's Guide* and the Resilient REST API documentation. You can access the object data using the keyword, `data_model`, and the respective FreeMarker syntax.

The following example assigns the incident object from the action data model to a variable called `incident`.

```
<#assign incident=.data_model.incident>
```

If you intend to have Resilient users fill in fields when performing a manual action, the Resilient administrator can create those fields and add them to the action dialog. You can then access the fields within the scripting context by accessing the action's properties, followed by the field's API access name.

For example, the following code defines the variables `mail_to`, `mail_cc`, and `mail_body`:

```
${.data.model.properties.mail_to}
${.data.model.properties.mail_cc}
${.data.model.properties.mail_body}
```

Note that different action types result in different data within the action data model. For example, `action.task` contains only data if the action is of type task. For more information about working with actions, see the *Resilient Incident Response Platform Master Administrator Guide*.

4.1.5. Additional Methods

As mentioned previously, you can apply all the FreeMarker template engine functionalities within the template's context (see the [FreeMarker manual](#)). In order to provide support for interaction with the Resilient platform, a few methods have been added to the template context.

- **REST Client to access the Resilient platform**

By using the keyword, `restClient`, you can access a representation of the Resilient Systems `SimpleClient` Java class. This class provides methods to execute HTTP requests to the Resilient platform REST API and uses the Resilient account that is connected with the IRHub.

The following is an example of how to post a note to the respective incident from within the template context:

1. Create the note data object:

```
<#assign note={"text": {"content":<p>Some HTML text</p>,"format":"html"}}>
```

2. Post the note using `restClient`:

```
<#assign note=restClient.post("incidents/" + incident.id?c + "/comments", note)>
```

For more details on using the Resilient REST API and `SimpleClient` class, see the `co3-api` project in the Resilient Systems GitHub repository.

- **Generating and adding unique ID's to an incident**

It might be desired to add a unique identifier to the outgoing email in order to be able to associate the email with an incident. The existing incident id might not always be ideal for this due to security concerns. Therefore, there are two methods within the template context that can help with creating ID's and adding them to the incident. Both methods use the configuration property `uuid_field_name` in order to update the field in the incident with the generated ID.

Random UUID method

The `getIncidentUuid` method expects the Resilient incident ID as an argument and generates a random UUID using the `java.util.UUID.randomUUID()` method. It posts the generated ID to the incident field, which is set in the `uuid_field_name` property, and returns it afterwards.

For example:

```
<#-- get incident UUID -->
<#assign uuid= getIncidentUuid(1234)>
```

SHA1 Hash method

Similar to the previous method, but it expects two arguments. First argument is again the Resilient incident ID and the second argument is the desired length of the generated SHA1 hash. It uses the incident name + incident description + current milliseconds since epoch in order to generate a SHA1 hash. The library used is `com.google.common.hash.Hashing` and `SHA1 hashString()` method. Furthermore, it also depends on the `uuid_field_name` property in order to update the desired incident field with the hash.

The following example generates and updates the incident with a SHA1 hash of length 10:

```
<#-- get incident UUID -->
<#assign hash=getIncidentSha1(incident.id?c, 10)>
```

4.2. Creating Actions

Once the template is available, the Resilient master administrators can create actions to send email. The administrator can create automatic and manual actions; however, the actions must use the same queue as defined in the `smtpservice.irhub.cfg` file.

When creating an action to send email using the Email Outbound Connector, the administrator needs to know the following:

- **Queue name.** This was defined during installation and based on the programmatic name of the message destination from the Resilient platform.
- **Template name.** As described in [File Naming Convention](#), the administrator must name the action based on the template name.

For more information about creating actions, see the *Resilient Incident Response Platform Master Administrator Guide*.

4.3. Adding a Certificate to the Keystore

In case of an encrypted connection to the email server, the certificate of the email server must be trusted. If you have installed the Email Connector, enter the same keystore file and password used by the Email Connector. If you did not install the Email Connector, you need to manually trust a certificate as follows:

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 smtpservice.irhub |grep keystore
```

3. Log out from IRHub and 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. It is recommended to use the existing Java keystore within IRHub `/usr/share/irhub/etc/irhub.truststore`.

```
$ 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 service irhub restart
```

4.4. Configuring Bi-Directional Email

As mentioned previously, you can use the Email Outbound Connector in conjunction with the Email Connector to allow for bi-directional email communication. This section provides an overview of the required setup and assumes that Email Connector and Email Outbound Connector are installed properly.

- **Email Outbound Connector setup**

In order to associate an email with an incident, the outgoing email as well as the incident need to contain the same unique ID. Therefore, you can use the methods mentioned in [Additional Methods](#) to create and update ID's to an incident.

This example assumes that a `uuid_field_name` property has been set, which is updated in the incident by the `getIncidentSha1` method. The same hash is also set within the email subject:

```
<#-- get incident hash -->
<#assign hash = getIncidentSha1(incident.id?c, 10)>
<#assign subject = "Thank you for your email {Ticket " + hash + "}">
```

- **Email Connector setup**

When a recipient replies to the email by sending it to a mailbox monitored by the Email Connector, its parsing script has to contain parsing rules in order to parse out the unique ID from the emails subject and perform a query for the existing incident. Therefore, you can use the parsing script's Query section.

The following example parses out the ID from the email subject, and assumes that the incident field provided for associating emails with incidents is a custom field called **mail.uuid**. On a successful query, the incident loaded into the following `<Script>` context is the respective incident with all data included. In case of other emails that do not apply to the parsing in the query, an empty incident is loaded into the `<Script>` context and will get posted as a new incident to the Resilient platform afterwards.

```
<Query>
  <![CDATA[
    var hash = mail.subject.match(/.*Thank you for your email
{Ticket (\d{10})}/);
    if(hash != null){
      [{
        "method":"contains",
        "field_name":"properties.mail_uuid",
        "value": hash[1]
      }];
    } else {
      null;
    }
  ]]>
</Query>
```

For more information about parsing emails, see the *Resilient Incident Response Platform Email Connector Installation and Configuration Guide*.

4.5. 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 Email Outbound Connector. The following is a list of the basic commands you need to configure the Email Outbound Connector:

- To log into the IRHub, establish a command shell on the host computer (e.g., use an ssh client such as PuTTY) as user, **resadmin**, and type **irhub**.
- To view all the properties that are required by the Email Outbound Connector:

```
irhub> property-list -p smtpservice.irhub |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 SMTP-related properties alphabetically, enter the following command:

```
irhub> property-list -p smtpservice.irhub |grep smtp |sort
```

- Use the property-set command to set the properties individually. For example, the following command configures the SMTP server to use john.smith@example.com as the address in the From field if the emails sent:

```
irhub> property-set -p smtpservice.irhub smtp_from john_smith@example.com
```

- Start the Email Outbound Connector for the changes to take effect.

```
irhub> restart 'Outbound EMail Actions Server'
```

- To confirm that the Email Outbound Connector is installed:

```
irhub> list | grep 'Outbound EMail Actions Server'
```

- To see the log messages within IRHub:

```
irhub> display
```

5. Uninstallation

To uninstall, remove the following files from IRHub:

```
/usr/share/irhub/deploy/irhub-smtp-service-kar-x.x.x.kar  
/usr/share/irhub/etc/MailHtmlTemplate.ftl  
/usr/share/irhub/etc/smtpservice.irhub.cfg
```

In addition, remove any HTML template files that were created for the Email Outbound Connector.

6. Troubleshooting

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

6.1. Managing the Log File

The pertinent log file for the Email Outbound Connector 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.

To view the log, log in to the IRHub and enter the log:display or log:tail command.

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

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
```

You can also change the log level specific to packages:

```
irhub>log:set DEBUG com.my.package
```

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

6.2. 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.

6.3. Useful Commands

Use the following command to verify that you can send email using SMTP from the host system. Once connected, you can issue standard SMTP commands to send email.

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

End of document