IBM Resilient

Resilient

Incident Response Platform

Resilient Incident Response Platform Integration Server Guide

Platform Version	Publication	Notes
32.0	January 2019	Initial publication.

Table of Contents

1. Introduction

An *integration server* is the system that you use to deploy integration packages, such as functions, to the Resilient platform.

This guide provides the procedures to create a Resilient integration server.

2. Architecture

The integration server must be able to connect to the Resilient platform. The Resilient platform can reside in your own environment (known as an on-premises configuration) or within the IBM cloud (known as a SaaS configuration).

2.1. Types of Integrations

The following types of integrations require an integration server:

- Python-based integrations, used by Resilient functions and Python-based custom actions. The Resilient platform can be in an on-premises or SaaS configuration. The integration server communicates with port 65001 of the platform using the STOMP messaging protocol.

- Java-based integrations, used by Java-based custom actions. The Resilient platform can be in an on-premises or SaaS configuration. The integration server communicates with port 65000 of the platform using ActiveMQ OpenWire.

- Custom threat services. The Resilient platform must be on-premises. The Resilient platform communicates with the integration over HTTPS on a port that you choose (by default on port 9000).

Specific integrations may have additional requirements, as described in the integration's documentation.

NOTE: There are also integrations in the form of plugins, such as for Splunk and QRadar, which escalate incidents to the Resilient platform using the REST APIs. You do not need a separate integration server for these plugins.

The following network diagram is an example of an integration server used for python-based integrations and a Resilient platform in a SaaS configuration. Resilient users access the platform from their web browsers.
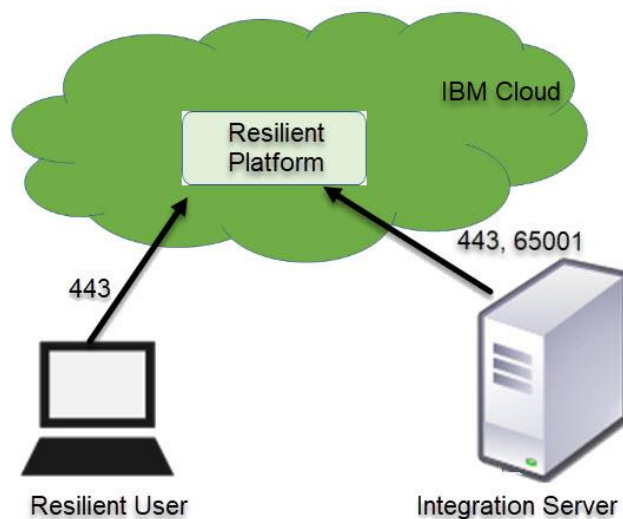
2.2. Network Configuration

If you have a Resilient platform in your environment, the integration server can be the same system as the one hosting the Resilient platform; however, IBM Resilient recommends that you keep them on separate systems for the following reasons:

- Security. Although all apps from the Community App Exchange are code scanned and tested, it is a security measure to not allow apps that are built in-house to have access to the command line of the Resilient platform host.

- Performance. Although the Resilient integrations are resource efficient, running them on the Resilient host could have impact on the available resources, especially if your integrations are heavily using in-memory processing.

- Access. The administrator of the integration server maintains, installs, and configures integrations. This administrator does not require access to the Resilient platform.

If the Resilient platform is beyond a firewall, such as in a SaaS configuration, configure the firewall to allow access to the following ports:

- 443. Required for the integration to connect to Resilient data using the REST API.
- 65000. Only if supporting Java-based integrations.
- 65001. Only if supporting Python-based integrations.
- 9000. Only if supporting custom threat feeds, which is not applicable for a Resilient platform in a SaaS configuration. This is "inbound" to the integration server, where the other connections are "outbound" from the integration server to the Resilient platform.

The integration server must also be able to access the hosts of the third-party applications that are integrated with the Resilient platform.

If you are planning to use your own custom threat services, you cannot deploy them to a Resilient platform in a SaaS configuration. The platform must be in your environment.

3. Prerequisites

The resources required by the integration server are variable due to the requirements of the integrations installed. Some integrations that operate on files in memory may have additional memory requirements. Integrations that perform decryption tasks may need more CPU. Therefore, IBM Resilient recommends the following prerequisites.

- Operating system:

  - o   Red Hat Enterprise or Centos Linux 7.4.
  - o   Windows (Windows Enterprise Server 2016 or later is recommended)

- For Python-based integrations, Python 2.7.*x* (where *x* is 9 or later), or 3.6.*x* (where *x* is 4 or later). Make sure the Python installation includes pip. On a Linux system, you can verify the version of Python using the following command:

  ```
  python --version
  ```

- Minimum of 5GB free disk space.
- Minimum of 8 GB RAM.
- Dedicated operating system account for running integrations.
- Text editor, such as nano, to edit the configuration file.
- Access to the Resilient platform. If the platform is beyond a firewall, configure the firewall to allow access to ports 443, 65000 if supporting Java-based integrations, 65001 if supporting Python-based integrations, and 9000 if supporting custom threat feeds.

The following lists the prerequisites for the Resilient platform:

- Version 30 or later.
- Licensed for Resilient Action Module. If using custom threat services, the platform must also be licensed for the Resilient Security Module. You can verify this by running the following command on the Resilient platform:

  ```
  resutil license
  ```

- Configured with a dedicated Resilient account to use as the API user. With most integrations, the account must have the permission to view and edit incidents, and view and modify administrator and customization settings.

- The Resilient Circuits components run as an unprivileged user, typically named integration. If you do not already have an integration user configured on your appliance, create it now.

- IBM Resilient recommends that you have a test environment. This can be a separate Resilient platform or a Resilient organization within your platform dedicated for testing.

4. Installation

To create an integration server, install the Resilient Circuits framework, which includes its dependent modules. These procedures assume that the integration server has access to the Internet. If not, see the section, Offline Installation.

4.1. Installing on a Linux System

The Resilient Circuits components run as an unprivileged user, typically named **integration**. If you do not already have an integration user configured on your system, create it now.

Install the Resilient Circuits framework as follows:

1. Use the ssh to access the command line interface on the integration server.

2. Install Resilient Circuits using the following command. This command also installs its dependent modules.
   ```
   pip install --upgrade resilient-circuits
   ```

3. Use the following command to verify that the Python modules, `resilient` and `resilient-circuits`, are installed.
   ```
   pip list
   ```

4. Configure Resilient Circuits as follows:

   a. Auto generate the app.config file as follows. This creates a directory, **.resilient**, in your home directory with a file in it called app.config, which is the default and preferred option. The Resilient Circuits configurations are maintained in the app.config file.
   ```
   resilient-circuits config -c
   ```

   The output of the command shows the directory where it installed the config file. By default, this directory is:
   ```
   /home/integration/.resilient/app.config
   ```

   If you require the configuration file to be in a different location or have a different name, you need to store the full path to the environment variable, APP_CONFIG_FILE.
   ```
   resilient-circuits config -c /path/to/<filename>.config
   ```

   b. Open the app.config file in your text editor. If using nano, the command would be:
   ```
   nano /home/integration/.resilient/app.config
   ```

   c. Replace the contents with the following settings. Your actual path names may be different.
   ```
   [resilient]
   host=localhost
   port=443
   email=resilient_account@example.com
   password=ResilientPassword
   org=Dev
   # componentsdir=/home/resadmin/.resilient/components
   logdir=/home/resadmin/.resilient
   logfile=app.log
   loglevel=INFO
   ```

   The **email** and **password** settings are for the Resilient account that you are using for integrations. Use the actual Resilient user name (in the form of an email address) and password.

   Use the actual Resilient organization name for the **org** name. It is not needed if the user account does not belong to more than one Resilient organization.

See [Updating the Configuration File](#) for a detailed description of all the app.config settings, especially **cafile** if your Resilient platform does not have a valid certificate.

    d.   Save the file.

    e.   Test your installation by running the following command:

```
resilient-circuits run
```

You are ready to download and deploy integration packages.

## 4.2. Installing on a Windows System

Resilient Circuits can be configured to run as a service on Windows. It requires the pywin32 library, which should be downloaded from [sourceforge](#). Instructions for downloading and installing the correct package are at the bottom of the sourceforge web page and must be followed carefully. Do not use the pypi/pip version of pywin32.

Installation of the wrong version of the pywin32 library will likely result in a Resilient service that installs successfully but is unable to start.

Enter the following command:

```
resilient-circuits.exe service install
```

Once installed, you can update the service to start up automatically and run as a user account.
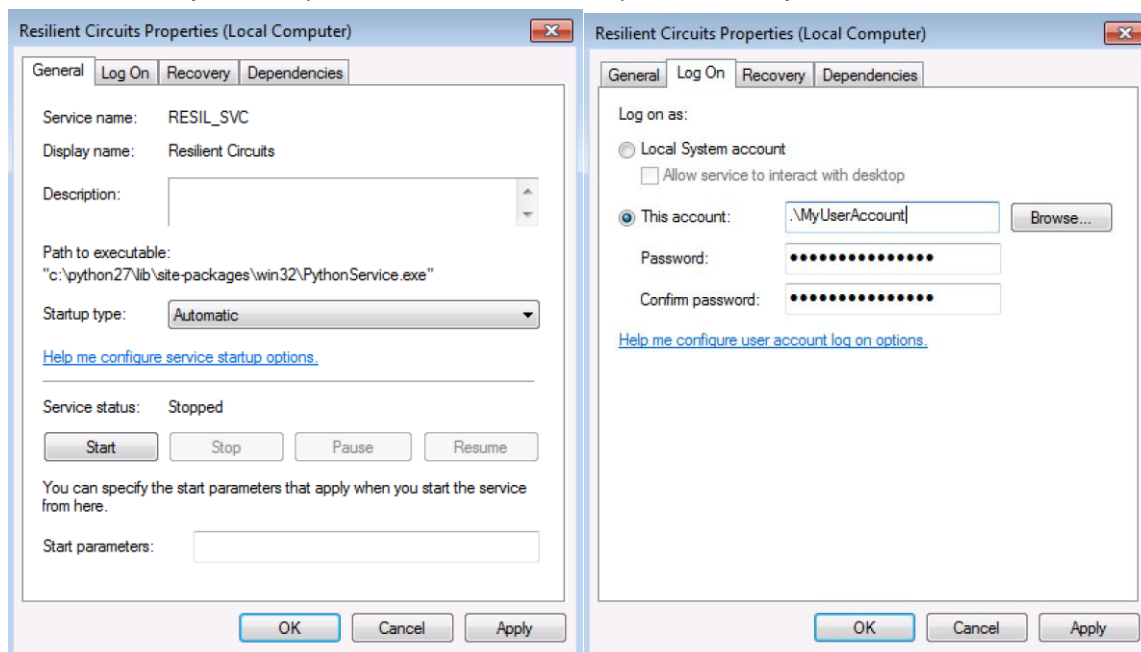
It is recommended that you log in as whichever user account the service will run as to generate the config file and confirm that the integration runs successfully with "resilient-circuits.exe run" before starting the service.

Commands to start, stop, and restart the service are provided as well.

```
resilient-circuits.exe service start
resilient-circuits.exe service stop
resilient-circuits.exe service restart
```

4.3. Configure Resilient Circuits for Restart

For normal operation, Resilient Circuits must run <u>continuously</u>. The recommend way to do this is to configure it to automatically run at startup. On a Red Hat appliance, this is done using a systemd unit file such as the one below. You may need to change the paths to your working directory and app.config.

1. The unit file must be named `resilient_circuits.service`. To create the file, enter the following command:

    ```
    sudo vi /etc/systemd/system/resilient_circuits.service
    ```

2. Add the following contents to the file and change as necessary:

    ```
    [Unit]
    Description=Resilient-Circuits Service
    After=resilient.service
    Requires=resilient.service
    [Service]
    Type=simple
    User=integration
    WorkingDirectory=/home/integration
    ExecStart=/usr/local/bin/resilient-circuits run
    Restart=always
    TimeoutSec=10
    Environment=APP_CONFIG_FILE=/home/integration/.resilient/app.config
    Environment=APP_LOCK_FILE=/home/integration/.resilient/resilient_circuits.lock
    [Install]
    WantedBy=multi-user.target
    ```

3. Ensure that the service unit file is correctly permissioned, as follows:

    ```
    sudo chmod 664 /etc/systemd/system/resilient_circuits.service
    ```

4. Use the systemctl command to manually start, stop, restart and return status on the service:

    ```
    sudo systemctl resilient_circuits [start|stop|restart|status]
    ```

You can view log files for systemd and the resilient-circuits service using the journalctl command, as follows:

```
sudo journalctl -u resilient_circuits --since "2 hours ago"
```

4.4. Offline Installation

If your integration server is not connected to the internet, you need to download the Resilient Circuits package from [IBM Resilient Github](#).

The following is an example transfer procedure for a Linux system.

```
mkdir ~/package_name_offline
pip download package_name -d "~/some_package_offline"
tar -cvfz package_name_offline.tar package_name_offline
```

Transfer the package to the integration server then perform the following:

```
tar -xzvf package_name.tar
cd some_directory
pip install package_name-x.x.x-py2.py3-x-x.whl -f ./ --no-index
```

5.   Updating the Configuration File

The configuration file defines essential configuration settings for all Resilient Circuits components running on the system.

By default, every Resilient integration package you install uses the same configuration file. If you need multiple 'resilient-circuits' applications running with different configuration files, you can set the APP_CONFIG_FILE environment variable to a different path for each process. For Linux, use the following command:

```
resilient-circuits config -c /path/to/<filename>.config
```

If APP_CONFIG_FILE is not set, the application looks for a file called "app.config" in the local directory where the run command is launched.

The [resilient] section of the configuration file controls how the core Resilient Circuits and Resilient packages access the Resilient platform. The following table describes all the required and optional values that can be included in this section.

NOTE: If on a Windows system and you edit the file with Notepad, please ensure that you save it as type **All Files** to avoid a new extension being added to the filename, and use UTF-8 encoding.

Parameter	Required?	Description
logfile	N	Name of rotating logfile that is written to logdir. Default is app.log.
logdir	N	Path to directory to write log files. If not specified, program checks environment variable DEFAULT_LOG_DIR for path. If that is not set, then defaults to a directory called "log" located wherever Resilient Circuits is launched.
log_level	N	Level of log messages written to stdout and the logfile. Levels are: CRITICAL, ERROR, WARN, INFO (default), and DEBUG.
host	Y	IP or hostname for the Resilient platform.
org	Y, if multiple orgs	Name of the Resilient organization. This is required only if the user account is used with more than one Resilient organization.
email	Y	User account for authenticating to the Resilient platform. It is recommended that this account is dedicated to integrations.
password	Y	Password for the Resilient user account.
no_prompt_password	N	If set to False (default) and the "password" value is missing from this config file, the user is prompted for a password. If set to True, the user is not prompted.
stomp_port	N	Port number for STOMP. Default is 65001.
componentsdir	N	Path to directory containing additional Python modules. Resilient Circuits can load custom components from this directory. Typically this option is only used by integration developers.
noload	N	Comma-separated list of: a. Installed components that should not be loaded. b. Module names in the componentsdir that should not be loaded. Example: my_module, my_other_module, InstalledComponentX

Parameter	Required?	Description
proxy_host	N	IP or Host for Proxy to use for STOMP connection. By default, no proxy is used.
proxy_port	N	Port number for Proxy to use for STOMP connection. By default, no proxy is used.
proxy_user	N	Username for authentication to Proxy to use for STOMP connection. If a proxy_host is specified and no proxy_user specified, then assumed no authentication is required.
proxy_password	N	Password for authentication to Proxy to use for STOMP connection. Used in conjunction with proxy_user.
cafile	Y	Path and file name of the PEM file to use as the list of trusted Certificate Authorities for SSL verification when the Resilient platform is using untrusted self-signed certificates. If not using a trusted certificate, **cafile** must be set to False. If there is a PEM file, use a second instance of **cafile** to set to True or False. If set to False, certificate verification is not performed and the PEM file is used. If set to True (default), allow only trusted certs.

6. Installing Integration Packages

Once Resilient Circuits is installed and running, you can install integration packages on your server. Whenever you install a new package, you need to update your app.config file to include any required sections for the new components.

You can download functions and other integration packages from the IBM Resilient Community or IBM X-Force App Exchange. Once downloaded, perform the following to install and configure the package on your integration server.

Note to Windows Users: To run integration commands on a Windows system, use resilient-circuits.exe. For example, "resilient-circuits.exe run" rather than "resilient-circuits run".

Use the following procedure to install integration packages in a tar.gz format.

1. Install your chosen package by first unzipping the file then using the following command:
    ```
    pip install <package_name>-x.x.x.tar.gz
    ```

2. Verify that the component is installed using the following command.
    ```
    resilient-circuits list
    ```

3. After installing the package, run the following command. This command adds a new section with default values in the app.config file for each package that was installed since the last update.
    ```
    resilient-circuits config –u
    ```

    You can choose to update specific packages:
    ```
    resilient-circuits config –u –l <package1> <package2>
    ```

    If using an alternate file location for your app.config file, you need to specify it when you update.
    ```
    resilient-circuits config –u /path/to/app.config
    ```

4. Follow the instructions in the component's documentation file to edit the app.config file. Depending on the requirements of the integration, you may need to modify the default values to fit your environment, such as credentials to a 3rd party system.

Files in a .res format contain components, such as scripts, workflows, and custom fields, which can be imported into your Resilient platform using the import procedure, as described in the *Resilient Incident Response Platform System Administrator Guide*.