



**Data Feed Integration Guide v1.0**

Incident Response Platform

Licensed Materials – Property of IBM

© Copyright IBM Corp. 2010, 2019.  All Rights Reserved.

US Government Users Restricted Rights: Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Resilient Incident Response Platform Data Feed Integration Guide

|  |  |  |
| --- | --- | --- |
| Version | Publication | Notes |
| 1.0 | March 2019 | Initial release. |

Table of Contents

[1. Overview 5](#_Toc3464206)

[1.1. Architecture 5](#_Toc3464207)

[1.2. Initial Population 6](#_Toc3464208)

[1.3. Useful Tools 6](#_Toc3464209)

[2. Prerequisites 7](#_Toc3464210)

[3. Installation 7](#_Toc3464211)

[3.1. Install the Python components 7](#_Toc3464212)

[3.2. Configure the Python components 7](#_Toc3464213)

[3.3. Deploy customizations to the Resilient platform 8](#_Toc3464214)

[3.4. Run the integration framework 8](#_Toc3464215)

[3.5. Configure Resilient Circuits for restart 9](#_Toc3464216)

[3.6. Confirm deployment 9](#_Toc3464217)

[4. Configuration 10](#_Toc3464218)

[4.1. Resilient Circuits config file 10](#_Toc3464219)

[4.1.1. ODBCFeed class 11](#_Toc3464220)

[4.1.2. FileFeed class 11](#_Toc3464221)

[4.1.3. SQLiteFeed class 12](#_Toc3464222)

[4.1.4. Additional connection strings 12](#_Toc3464223)

[4.1.5. Database Field Length Considerations 12](#_Toc3464224)

[4.1.6. Additional considerations 12](#_Toc3464225)

[4.2. Resilient platform configuration 13](#_Toc3464226)

[4.2.1. Rules for data tables 14](#_Toc3464227)

[5. Test 14](#_Toc3464228)

[6. Troubleshoot 15](#_Toc3464229)

[7. Support 15](#_Toc3464230)

# Overview

This guide describes the Resilient Data Feed extension. This extension allows you to maintain "replica" data from a Resilient platform for access by other tools, such as Business Intelligence solutions. The replications and updates are performed in near real-time. The data written represents the current state of incident data, notes, artifacts, data tables, and so on.

## Architecture

This extension allows you to perform queries on Resilient data without having to access the Resilient database directly. You can then run business intelligence queries from other tools.

The data can be written to any or all of the following destinations:

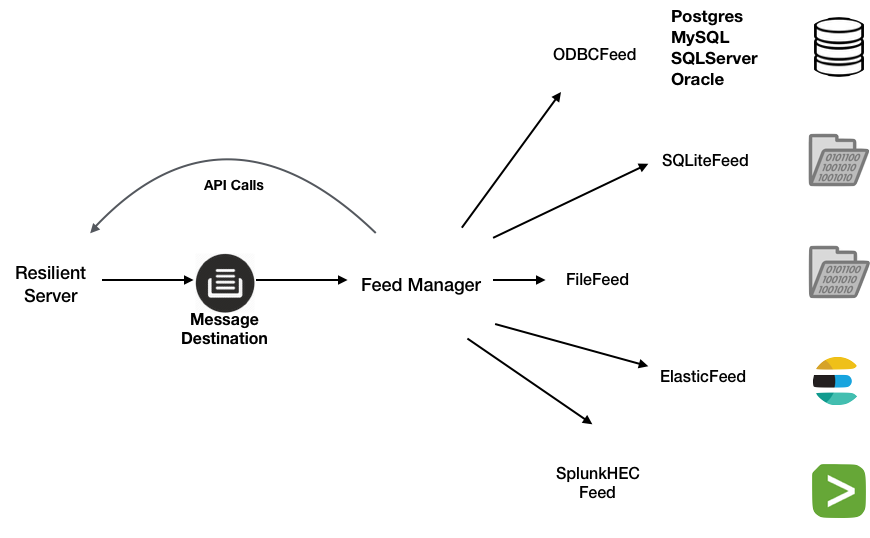
* ODBC databases:
  + PostgreSQL
  + MySQL (MariaDB)
  + Microsoft SQLServer
  + Oracle
* SQLite file/database
* ElasticSearch
* Splunk ES using the HTTP Event Collector
* Local directory (one file per object)

The content is consistent with the Resilient type/field semantic model and includes custom fields. For SQL destinations, the table name is the same as the type name (such as incident, task, and artifact) and the column names are the same as the field "programmatic name".

Data tables are supported (the programmatic name will be the same as the DB table name).

Newly added fields and data tables are added in the existing destination database.

The architecture allows other destinations to be created with relative ease. The following diagram shows the overall data flow from the Resilient platform to the existing feeds.



## Initial Population

When you run the Data Feed extension against a Resilient platform that has pre-existing data, it can optionally read all that platform’s data and populate your feeds. After the initial data population, the extension performs all updates by listening on a message destination (queue) from which integration logic is called to update the data source.

## Useful Tools

Most "business intelligence" tools allow you to query data from SQL databases. Here are some tools that you can use to run queries and reports against this data:

* Tableau
* Microsoft Power BI
* IBM Cognos
* Grafana
* Interactive SQL, for example:

-- Number of artifacts for each incident

select i.id, i.name, count(\*) num\_artifacts

from incident i join artifact a on i.id = a.inc\_id

group by i.id, i.name;

# Prerequisites

Before installing, verify that your environment meets the following prerequisites:

* Resilient platform is version 31 or later.
* You have access to a Resilient integration server. An *integration server* is the system that you use to deploy integration packages to the Resilient platform. See the [Resilient Integration Server Guide (PDF)](https://github.com/ibmresilient/resilient-reference/blob/master/developer_guides/Integration%20Server%20Guide.pdf) for more information.
* The following environments have been tested and are recommended:
  + PostgreSQL 9.6 or higher
  + MySQL 5.7 (MariaDB 10.3) or higher
  + Microsoft SQL Server 2017
  + Oracle 12c
  + Splunk ES 7.1
  + ElasticSearch 7.0

# Installation

The integration package contains Python components that are called by the Resilient platform. These components run in the Resilient Circuits integration framework. The package also includes Resilient customizations that will be imported into the platform later.

You perform these installation procedures at the Resilient integration server.

## Install the Python components

Complete the following steps to install the Python components:

1. Ensure that the environment is up-to-date, as follows:

sudo pip install --upgrade pip

sudo pip install --upgrade setuptools

sudo pip install --upgrade resilient-circuits

1. Run the following command to install the package:

sudo pip install --upgrade fn\_data-feed-1.0.0.zip

## Configure the Python components

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.

Complete the following steps to configure and run the integration:

1. Using sudo, switch to the integration user, as follows:

sudo su - integration

1. Use one of the following commands to create or update the resilient-circuits configuration file. Use –c for new environments or –u for existing environments.

resilient-circuits config -c

or

resilient-circuits config –u [-l rc\_data\_feed]

1. Edit the resilient-circuits configuration file, as follows:
   1. In the [resilient] section, ensure that you provide all the information required to connect to the Resilient platform.
   2. In the [feeds] section, define the feeds you intend to use and create separate sections for each feed. For example:

[feeds]

feed\_names=my\_postgresql\_feed,my\_file\_feed,my\_sqlite\_feed

reload=True

# feed\_data is the default queue that will be listened to

queue=feed\_data

[my\_postgresql\_feed]

class=ODBCFeed

odbc\_connect=Driver={PostgreSQL};Server=localhost;Port=5432;Database=feed

sql\_dialect=PostgreSQL96Dialect

uid=mypguser

pwd=mypassword

[my\_file\_feed]

class=FileFeed

directory=/var/resilient/feed\_data

[my\_sqlite\_feed]

class=SQLiteFeed

file\_name=/var/resilient/feed.sqlite3

See the [Configuration](#_Configuration) section for details.

## Deploy customizations to the Resilient platform

The package contains rules required for the flow of incident data changes to the data feeds.

1. Use the following command to deploy these customizations to the Resilient platform:

resilient-circuits customize [-l rc\_data\_feed]

1. Respond to the prompts to deploy the message destination and rules.

## Run the integration framework

To test the integration package before running it in a production environment, you must run the integration manually with the following command:

resilient-circuits run

The resilient-circuits command starts, loads its components, and continues to run until interrupted. If it stops immediately with an error message, check your configuration values and retry.

## Configure Resilient Circuits for restart

For normal operation, Resilient Circuits must run continuously. 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

1. 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

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

sudo chmod 664 /etc/systemd/system/resilient\_circuits.service

1. 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"

## Confirm deployment

Once the package deployment is complete, you can view them in the Resilient platform Rules tab, as shown below.



### Configuration of Rules for data tables

Data tables are also supported by the Data Feeder. Because rules are written to specific data tables, they require a Resilient administrator to create them manually. The following screenshot shows a rule which references a data table and it triggers for any row change (insert, update, delete). Follow this procedure for all the data tables you need to include in the Data Feeder.



# Configuration

There are two aspects to configuration of the Data Feed extension: Resilient Server configuration and app.config configuration.

## Resilient Circuits config file

The configuration goes in the standard app.config file. There is a "feeds" section that contains a "feed\_names" key, which is a comma separated list of feeds that you want to enable. The names you use represent the names of the subsequent sections that contain the actual feed configurations. Each name must be unique. For example:

[feeds]

feed\_names=my\_postgresql\_feed,my\_file\_feed,my\_sqlite\_feed

reload=True

# feed\_data is the default queue that will be listened to

queue=feed\_data

[my\_postgresql\_feed]

class=ODBCFeed

odbc\_connect=Driver={PostgreSQL};Server=localhost;Port=5432;Database=feed

sql\_dialect=PostgreSQL96Dialect

uid=mypguser

pwd=mypassword

[my\_file\_feed]

class=FileFeed

directory=/var/resilient/feed\_data

[my\_sqlite\_feed]

class=SQLiteFeed

file\_name=/var/resilient/feed.sqlite3

In this example feed\_names references three feeds. The comma separated names refer to additional sections of configuration settings.

This structure allows you to create as many feeds as you need, regardless of the "class". For example, you can have three different ODBC feeds as long as they all have unique names.

You can perform an initial data population by setting reload=True in the app.config [feeds] section and then restarting Resilient Circuits. Be aware that every time Resilient Circuits starts with reload=True, the entire set of Resilient incidents, notes, artifacts, and so on are refreshed in your feeds.

The following sections explain the configuration options for each of the feed classes.

## ODBCFeed class

The ODBCFeed class is probably the most flexible and useful of the feeds. It allows you to write all the incoming data to an ODBC database.

The following configuration items are supported:

| Key | Values | Description |
| --- | --- | --- |
| class | ODBCFeed | Indicates that the section is for an ODBCFeed. |
| odbc\_connect | ODBC connect string | Example for PostgreSQL: Driver={PostgreSQL};Server=localhost;Port=5432;Database=feed |
| sql\_dialect | PostgreSQL96Dialect,  MariaDBDialect,  SQLServerDialect,  OracleDialect | Name of the SQL dialect. |
| uid | DB user name | Specify the database user name in this property and not in the connect string. Most DBs support the uid in the connect string but you should specify in this property instead. |
| pwd | DB password | Specify the database user's password in this property and not in the connect string. Most DBs support the pwd in the connect string but you should specify it in this property instead. You can use the standard Resilient Circuits mechanism for secure password storage. |

When using a data feed database**, IBM Resilient strongly recommends that you create and maintain** the database on system separate from the Resilient platform, where queries cannot impact your running Resilient instance. Allowing access to the Resilient platform for a database instance can also compromise security of the platform itself.

### Additional connection strings

The following tables lists additional database connection strings for the other supported databases.

|  |  |
| --- | --- |
| Database | Connection Strings |
| MariaDB | Driver={MariaDB ODBC 3.0 Driver};Server=127.0.0.1;DB=;Port=3306;connectTimeout=0 |
| Oracle | Driver={Oracle 12c ODBC driver};DBQ=ORCLCDB |
| SQLServer | DRIVER={ODBC Driver 17 for SQL Server};SERVER=127.0.0.1;PORT=1443;DATABASE=<yourDB>; |

### Database Field Length Considerations

Each database type has limits to the size of data stored. The following table describes the limits for each database.

|  |  |  |
| --- | --- | --- |
| Database | Field | Limit |
| Postgres | text | 1GB |
| MySQL/MariaDB | text | 4GB |
| MS SQLServer | varchar(max) | 2GB |
| MS SQLServer | varchar(xx) | 4000 |
| Oracle | varchar2(xx) | 8000 |

Some databases support blobs which can be supported specific cases. The disadvantage to blobs is that they are unsearchable.

### Additional considerations

Some databases have reserve words which cannot be used in tables (such as date and size). If a Resilient custom field is found to be in a database reserve list, the name is altered to include a trailing '\_' (for example, the column name).

## FileFeed class

The FileFeed class allows you to write all the incoming data to a directory on your file system (one file per object). The structure of the file names is either:

incident\_{inc\_id}.json (for incident objects)

or (for all other types of objects):

incident\_{inc\_id}\_{type}\_{obj\_id}.json

In this context, "type" can be:

* task
* artifact
* milestone
* note
* data table programmatic name

The following configuration items are supported:

|  |  |  |
| --- | --- | --- |
| Key | Values | Description |
| class | FileFeed | Indicates that the section is for a FileFeed. |
| directory | Path for a directory on local system (required) | Location where the files are to be written. |

## SQLiteFeed class

The SQLiteFeed class allows you to write all the incoming data to a SQLite DB file. SQLite is very useful for testing and in cases where you want to have the data stored in a single file that you can easily share. Some tools may natively support SQLite as well.

SQLite supports CSV formatting, so you can easily export the data from the SQLite file into a CSV file, which can then be imported into another tool, such as Excel, for further analysis.

The following configuration items are supported:

|  |  |  |
| --- | --- | --- |
| Key | Values | Description |
| class | SQLiteFeed | Indicates that the section is for an SQLite. |
| file\_name | Path for a local file on the system where the SQLite DB resides. | This is created if it does not exist. If it does exist, it must be an SQLite database. |

## ElasticFeed

This class allows you to write all incoming data to ElasticSearch. The data representation within Elastic is referenced by index, document type (incident, note, task, artifact, etc.) and document\_id (incident\_id, note\_id, task\_id, etc.).

The following configuration items are supported:

|  |  |  |
| --- | --- | --- |
| Key | Values | Description |
| class | ElasticFeed | Indicates that the section is for an ElasticSearch. |
| url | Ex. https://eliastic.yourorg.com | URL of Elastic server. Port is specified in it’s own parameter. |
| port | Ex. 9200 | Default is 9200 |
| index | Ex. resilient | if using multiple organizations, consider indexes such as resilient<org\_ID> |
| auth\_user  auth\_password |  | User and password to authenticate to ElasticSearch. |
| cafile | True | False | Specify ‘false’ to bypass certification authentication |

### Considerations

ElasticSearch allows for the updating to and deleting of individual documents. No data duplication occurs.

## SplunkHECFeed

The SplunkHECFeed class utilizes the Splunk HTTP Event Collector for data import. This is convenient as the data from Resilient is readily converted to JSON which can be natively consumed by Slunk.

|  |  |  |
| --- | --- | --- |
| Key | Values | Description |
| class | SplunkHECFeed | Indicates that the section is for Splunk ES. |
| token | Ex. 81e2d4bb-c008-49ac-a7a7-c0bf408c999 | API token defined for the HEC data input. |
| host | Ex. splunk.yourorg.com |  |
| port | Ex. 8088 | The default is 8088 |
| index | Ex data\_feeder | The name of the index already defined. |
| event\_host | Ex. myorg.com | Optional host name to record as the source of the events. |
| event\_source=resilient | Ex. resilient | Optional source name of the events. Specifying a value improves searching |
| event\_source\_type |  | Optional source\_type if one value is used for all events. If unspecified, each object type (incident, task, note, etc.) is used as the source\_type |
| use\_ssl | True | False | Indicate if connections to the HEC uses encryption (https) |

### Considerations

Enable the HTTP Event Collector within Splunk ES before using this data feed.

Splunk events are immutable. Resilient object changes are represented as new events. No event deletion is possible.

# Test

***Are there any tests that users can/should run?***

# Troubleshoot

There are several ways to verify the successful operation of a Resilient extension.

* Resilient Action Status

When viewing an incident, use the Actions menu to view Action Status. By default, pending and errors are displayed. Modify the filter for actions to also show Completed actions. Clicking on an action displays additional information on the progress made or what error occurred.

* Resilient Scripting Log

A separate log file is available to review scripting errors. This is useful when issues occur in the pre-processing or post-processing scripts. The default location for this log file is: /var/log/resilient-scripting/resilient-scripting.log.

* Resilient Logs

By default, Resilient logs are retained at /usr/share/co3/logs. The client.log may contain additional information regarding the execution of functions.

* Resilient-Circuits

The log is controlled in the .resilient/app.config file under the section [resilient] and the property logdir. The default file name is app.log. Each function will create progress information. Failures will show up as errors and may contain python trace statements.

# Support

For additional support, contact [support@resilientsystems.com](mailto:support@resilientsystems.com).

Including relevant information from the log files will help us resolve your issue.