IBM Resilient

resilient

Incident Response Platform

SYMANTEC ICDX INTEGRATION GUIDE v1.0

Resilient Incident Response Platform Symantec ICDx Integration Guide

Version	Publication	Notes
1.0	January 2019	Initial release.

Table of Contents

1. Overview

This guide describes the Resilient extension for Integrated Symantec Cyber Defense Exchange (ICDx).

Symantec's Integrated Cyber Defense Exchange is an open platform to collect log and event information from many of the Symantec products, as well as others, in a normalized format according to a schema. You can enrich Resilient incidents with relevant information of potential threat events, as well as create new incidents through the orchestration and automation tools provided by this package.

The Integrated Cyber Defense Schema is an information model that organizes attributes, and the objects which are made up of attributes, into event types using a standard cross-product schema. The event types fall into a number of categories. ICDx normalizes event data coming from data collector products and aligns the data to attributes in the Integrated Cyber Defense Schema.

1.1. Use Case Scenarios

The ICDx integration supports the following general use cases:

- Gathering Event data by UUID. Users can get access to any event for any Symantec product which is configured with ICDx in a normalized format.
- Searching for events by criteria. Users can find specific events event for any Symantec product which is configured with ICDx in a normalized format. For example, users can query all events of type 8031 (File Detection).
- Searching for events contained within a specific archive. ICDx maintains the ability to isolate events within one or more created archives. The benefit of this is having a subset of event data which can be searched more efficiently. Users on the Resilient platform can create a workflow which executes a Get Archive List Function and then passes the results downstream to refine the Find Events query to search a specific archive only.
- Create incidents automatically. Users can setup the Forwarder Component to create incidents automatically in the background with relevant artifact data to reduce the time needed to report on these event alerts.

1.2. ICDx Forwarders and Collectors

The ICDx integration allows you to ingest data into your environment. The integration works with ICDx through Collectors and Forwarders.

A *Collector* is an ICDx component that enables you to specify where data is collected, such as other production systems. A *Forwarder* is the ICDx component that allows you to relay or forward information to a destination, such as another ICDx installation, a HTTP API or in this case to an AMQP component.

Forwarders allow you to define a query predicate then compare events as they are ingested against this predicate. Matching events are sent to a forwarder queue, where they remain until they are consumed.

For more information on how Collectors and Forwarders work, refer to the ICDx System Administration Guide.

1.3. Integration components

The integration package consists of the following components:

- ICDx: Get Event Function
- ICDx: Find Events Function
- ICDx: Get Archive List Function
- Example Workflows used to invoke the Functions
- A Data Table UI Component
- Standalone ICDx Forwarder

The functions included with this package enable you to use orchestration to gather enrichment artifacts or other actionable information whereas, the standalone ICDx forwarder provides automation capabilities to have this enrichment performed for you combined with incident creation.

1.3.1. ICDx: Get Event Function

The Get Event function retrieves a single event using an event UUID. By default, all attributes, including nested objects, are returned in the result.

Name	Type	Example	Mandatory
`ICDx_uuid`	String, unique identifier for an event	`“ec6167c0-1c2e-11e8-c000-000000000022”`	**Y**

If a matching event is found with the provided UUID, a response of 200 is returned in the headers of the AMQP request. If no result is found, the status is 204, indicating no found event.

Part of this function execution involves parsing the result payload for potential artifacts. Artifacts found are then formulated into a data object mapped to the relevant Resilient artifact name. The result payload includes a formatted dictionary of artifacts, parsed from the event, with their relevant Resilient artifact name to simplify artifact creation.

The example workflow provided with the integration includes a rule that allows users to run the Get Event functions on artifacts, taking in the artifact as the UUID input of type String. The rule could also be configured to work with custom artifact types with minimal changes.

Additionally, a second provided rule and workflow is intended to be run on a row of the ICDx Queried Events Data Table included with this package. The rule and workflow pulls the UUID value from this row. The intent of this is to enable you to search for events, where the results are populated into the data table and then any event of importance can be further queried using this function.

1.3.2. ICDx: Find Events Function

The Find Events function request retrieves events that are within the specified time range and satisfy a specified search condition. By default, all attributes, including nested objects, are returned from each event as a JSON dictionary. Note that the UUID attribute is always returned, even if it is not explicitly included in a defined list of fields to return.

Name	**Type**	**Example**	**Mandatory**

ICDx_search_request	JSON payload containing search criteria	`{`
		`    "id" : 1,`
		`    "start"  : "-24h",`
		`    "where"  : "type_id = 8031",`
		`    "fields" : ["time", "device_name",`
		`               "device_ip"],`
		`    "limit"  : 3`
		`}`
			Y

Making requests to the Find Events API requires at a JSON payload which, at a minimum, must include an ID attribute. When making a request, if a matching event or number of events are found with the provided search query, a response of 200 is returned in the headers of the AMQP Request. If no result was found, the status is 204, indicating no found events.

Results returned from this Find Events function includes a list of all matching events. The package includes four example workflows that demonstrate the usage of the Find Events function. In each case, the results of the search request are added into the ICDx Queried Events Data Table, provided with this package.

The example provided in the table searches for File Detection (type_id=8031) related events that have occurred within the last 24 hours. The search result is limited to the first three events found and of those three, only the time, device_name and device_ip attributes are returned.

1.3.3. ICDx: Get Archive List Function

The ICDx platform stores event data in a number of archives. These archives are collections of related information, such as System events. An archive can be setup as a dedicated archive for certain events. In the Find Events API request, the `from` attribute is a string array of the archives to be searched. Each array element is the relative path to the archive, as returned by the Get Archive List API.

The Get Archive List API allows a user to return an array of available archives including path information.

The information returned from the Get Archive List Function is quite useful for limiting which archives are used for a Find Events Function. An example workflow, provided with this package, demonstrates using this function to return the available archives and then use the results to search specific archives.

1.3.4. Resilient Forwarder component

This is a Resilient Circuits component that is configured to read from an ICDx Forwarder. The purpose of this component is to ingest forwarded event data from your ICDx installation and then use this information to create incidents in the Resilient platform. The Resilient Forwarder attempts to perform automatic artifact enrichment for each incident created by parsing for potential artifacts and formatting the potential artifact data to align with how the Resilient platform describes artifacts.

The Resilient Forwarder configures an AMQP connection, similar to the provided functions. However, instead of connecting to the Search API which uses the `dx.archives.search` queue, the Resilient Forwarder establishes a connection to a Forwarder queue.

The Resilient Forwarder is based on the same Circuits Component architecture as Resilient functions with the difference being, this component is not invoked from the execution of a rule, workflow or through the use of a message destination. Instead, the component has a configurable value in the app.config file which specifies whether this component is to be run when Resilient Circuits is run.

The Resilient Forwarder runs alongside Resilient Circuits but works in a separate background thread which avoids blocking any other functions from executing.

When an event is forwarded, the Resilient Forwarder prepares an IncidentDTO schema by parsing the event for enrichment data which is then sent to your Resilient platform through the REST API. The end result is one incident created per event forwarded.

2. Check Prerequisites

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

- Resilient platform is version 30 or later.
- You have a Resilient account to use for the integrations. This can be any account that has the permission to view and modify administrator and customization settings, and read and update incidents. You need to know the account username and password.
- 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) for more information.
- Open port for communication with ICDx. By default, this port is 5672. In the event the integration server is hosted separate to the Resilient platform per recommendations, you only need to open the ICDx message queue port on the integration server.

3. Install the Integration

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.

3.1. 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
```

2. Run the following command to install the package:

```
sudo pip install --upgrade fn_icdx-1.0.0.zip
```

3.2. 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
```

2. 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
```

3. Edit the resilient-circuits configuration file, as follows:

   a. In the [resilient] section, ensure that you provide all the information required to connect to the Resilient platform.

   b. In the [fn_*icdx*] section, edit the settings as follows:

```
icdx_amqp_host = <YOUR_ICDX_HOST>

icdx_amqp_port = <YOUR_ICDX_PORT>

icdx_amqp_vhost = <YOUR_ICDX_HOST>

icdx_amqp_username = <YOUR_ICDX_USERNAME>

icdx_amqp_password = <YOUR_ICDX_PASSWORD>

icdx_forwarder_toggle = <True / False>

icdx_forwarder_inc_owner = <USER_EMAIL / USER_ID / GROUP_NAME>
```

   The `icdx_forwader_toggle` specifies whether the Forwarder is to run in the app.config.

The `icdx_forwarder_inc_owner` defines who is to be assigned the created incidents. This can be provided as a userID, a user's email address or an entire group within your Resilient platform.

3.3. Deploy customizations to the Resilient platform

This package contains six example rules, six example workflows triggered by the rules and three functions for interacting with the ICDx Search API.

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

   ```
   resilient-circuits customize
   ```

2. Respond to the prompts to deploy functions, message destinations, workflows and rules.

3.4. 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.

3.5. 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
   ```

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:

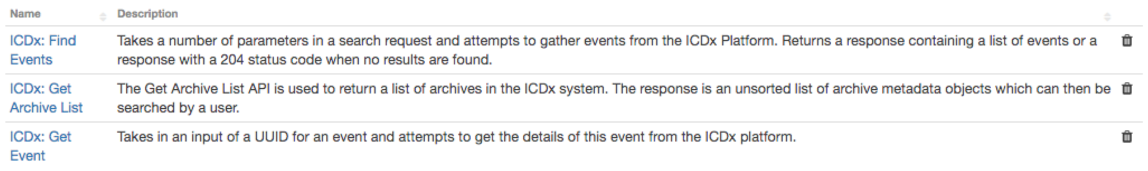
```
sudo systemctl resilient_circuits [start|stop|restart|status]
```

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"
```

3.6. Confirm deployment

Once the integration deploys, you can view the functions in the Resilient platform Functions tab, as shown below. The integration also includes example workflows and rules that show how the functions can be used. You can copy and modify these workflows and rules for your own needs.



Name	Description
ICDx: Find Events	Takes a number of parameters in a search request and attempts to gather events from the ICDx Platform. Returns a response containing a list of events or a response with a 204 status code when no results are found.
ICDx: Get Archive List	The Get Archive List API is used to return a list of archives in the ICDx system. The response is an unsorted list of archive metadata objects which can then be searched by a user.
ICDx: Get Event	Takes in an input of a UUID for an event and attempts to get the details of this event from the ICDx platform.

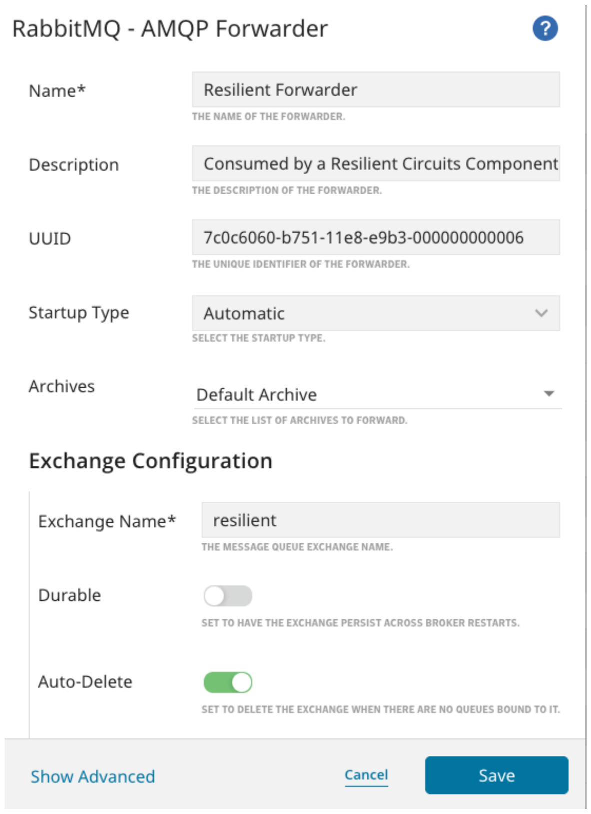
4. Configure ICDx

You need to configure your Symantec ICDx system to use the Resilient forwarder and enable automatic Incident creation.

It is important that ICDx and Resilient platform can communicate using AMQP. The default AMQP port for ICDx is port 5672. Make sure that the port ICDx uses is opened in the Resilient platform.

The ICDx Forwarders are available in a number of formats, including ElasticSearch, JSON and AMQP. This Resilient integration makes use of the AMQP Forwarder, which must be configured on your ICDx system with the following settings.

- **Message Queue Exchange Name**. An Exchange Name of `resilient` is required.
- **Durable**. Set the Durable toggle to false to disable persistence between broker restart
- **Auto-Delete**. Set an Auto-Delete value of true to delete the Exchange when no Message Queues are bound to it. This is done in ICDx using a toggle as follows:



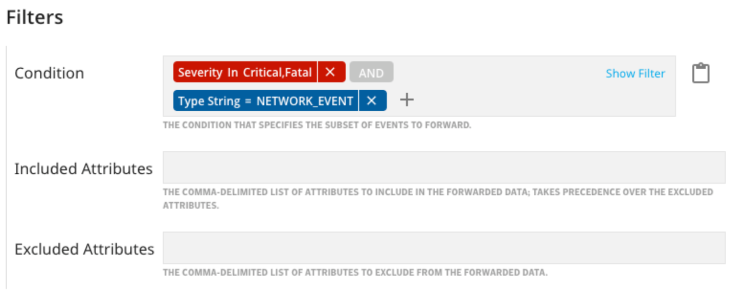
RabbitMQ - AMQP Forwarder

Name*	Resilient Forwarder THE NAME OF THE FORWARDER.
Description	Consumed by a Resilient Circuits Component THE DESCRIPTION OF THE FORWARDER.
UUID	7c0c6060-b751-11e8-e9b3-000000000006 THE UNIQUE IDENTIFIER OF THE FORWARDER.
Startup Type	Automatic SELECT THE STARTUP TYPE.
Archives	Default Archive SELECT THE LIST OF ARCHIVES TO FORWARD.

Exchange Configuration

Exchange Name*	resilient THE MESSAGE QUEUE EXCHANGE NAME.
Durable	off SET TO HAVE THE EXCHANGE PERSIST ACROSS BROKER RESTARTS.
Auto-Delete	on SET TO DELETE THE EXCHANGE WHEN THERE ARE NO QUEUES BOUND TO IT.

- **Condition Filter**. Using the default settings on a Symantec Forwarder can cause a lot of incident generation. You can filter incident generation to specific conditions using the ICDx Condition parameter.



Filters

Condition: Severity In Critical,Fatal AND Type String = NETWORK_EVENT

THE CONDITION THAT SPECIFIES THE SUBSET OF EVENTS TO FORWARD.

Included Attributes:

THE COMMA-DELIMITED LIST OF ATTRIBUTES TO INCLUDE IN THE FORWARDED DATA; TAKES PRECEDENCE OVER THE EXCLUDED ATTRIBUTES.

Excluded Attributes:

THE COMMA-DELIMITED LIST OF ATTRIBUTES TO EXCLUDE FROM THE FORWARDED DATA.

ICDx Conditions are compiled using a query language which supports most operators and allows you to define a strict query predicate on ICDx. This reduces the number of events sent. Alternatively, you may define logic on the Resilient platform which aggregates incidents, such as an automatic rule when an incident is created.

5. Inform Resilient Users

The following provides helpful information when using the ICDx Integration.

5.1. Search Requests and their Input Payloads

All functions provided in this integration take an input of a JSON payload. The aim of this is to provide the user with flexibility to form complex queries without ever leaving the Resilient platform.

Each function targets a different part of the API through AMQP. As a result of this, each type of request has its own unique ID attribute, which must be provided with the request payload to ensure the right endpoint is hit. This is because the AMQP Search API attempts to provide a similar interface to the HTTP REST API but at same time providing the benefits of a producer/consumer architecture.

5.2. Where vs Filter Conditions

ICDx events can be queried a number of ways. For a faster query, it is recommended to use the `where` attribute of the Search Request which enables you to search against a specific set of Indexed Attributes, improving search times. The default list of these indexed search attributes includes:

- category_id
- collector_name
- collector_uid
- device_ip
- device_name
- device_os_name
- event_id
- feature_name
- feature_ver
- id
- product_name
- product_ver
- severity_id
- type_id
- user_name

If you wish to compile a query which searches against attributes not outlined above, these search condition should be submitted as a part of the `filter` attribute. The `filter` attribute is used to filter out unwanted results and can be used to return a defined set of results which meet any number of conditions.

Search requests done with ICDx use the `where` attribute to return an initial set of results which are then filtered to return the final result.

6. Troubleshooting

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

- 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.

7. Support

For additional support, contact support@resilientsystems.com.

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