

Anomali Enrichments SDK

Enrichments SDK Guide

Release: 3.0

March 29, 2021

ANOMALI®

Copyright

Copyright © 2021 Anomali, Inc. All rights reserved.

Anomali is a registered trademark, ThreatStream is a registered servicemark, and Optic, Integrator, STAXX, and Anomali Enterprise are trademarks of Anomali, Inc.

All other brands, products, and company names used herein may be trademarks of their respective owners.

Support

Support Portal	https://support.anomali.com
Email	support@anomali.com
Phone	+1 844-4-THREATS
Twitter	@anomali

Documentation Updates

Date	Release	Description
3/29/2021	3.0	Updated for the 3.0 release.
9/9/2020	2.0	Added " Updating Enrichment Credentials " on page 47.
5/12/2020	2.0	Updated " Creating Enrichment Metadata Files " on page 32.
3/31/2020	2.0	Updated for the 2.0 release.

CONTENTS

About This Release	4
Chapter 1: Introduction	5
Chapter 2: Developing Context-Based Enrichments	9
Chapter 3: Developing Pivot-Based Enrichments	23
Chapter 4: Testing Your Python Script	29
Chapter 5: Creating Enrichment Bundles	32
Chapter 6: Testing ThreatStream Cloud Enrichment Bundles	40
Chapter 7: Submitting ThreatStream Cloud Enrichments for Certification	43
Chapter 8: Updating Enrichment Bundles	47
Chapter 9: Troubleshooting Your Enrichment	49
Appendix A: Sample Documentation	52

About This Release

What's New in 3.0

This release includes the following enhancement:

Python 3: The Anomali Enrichments SDK is now Python 3 compatible. You can use the Enrichments SDK to develop Python 3 based enrichments for ThreatStream Cloud. Note that enrichments built using Enrichments SDK v3.0 are not supported on ThreatStream OnPrem v5.1 and earlier or ThreatStream AirGap v4.1.1 and earlier.

Chapter 1: Introduction

The Anomali Enrichments Software Development Kit (SDK) includes python libraries and sample code that can be used to develop enrichments for ThreatStream Cloud. Once developed and activated, data enrichments from sources outside ThreatStream can be leveraged from observable details pages or the Explore pivoting tool.

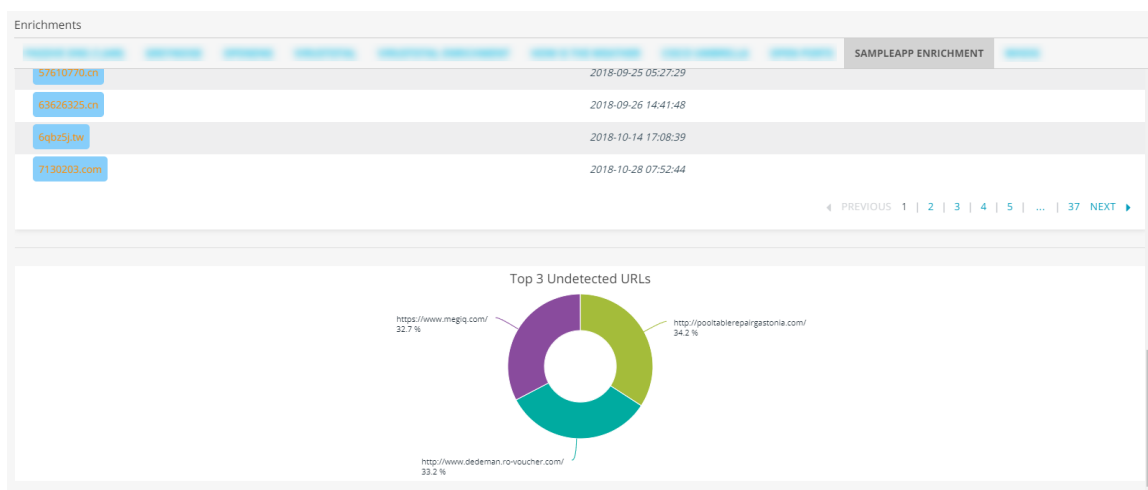
Note: The Anomali Enrichments SDK v3.0 uses Python 3. Python 3 enrichments are not supported on ThreatStream OnPrem v5.1 and earlier or ThreatStream AirGap v4.1.1 and earlier.

On ThreatStream Cloud, enrichments can be developed for use by your organization only or the entire Anomali Community.

There are two types of enrichments you can develop using the Anomali Enrichments SDK: context-based enrichments and pivot-based enrichments.

Context-Based Enrichments

Context-based enrichments enable analysts to step through entire workflows on observable details pages, thus eliminating the need to jump between product screens. Up to 50 observables from the enrichment source can be displayed. The Anomali Enrichments Library enables you to customize how data is displayed on observables details pages. The following is an example of a context-based enrichment:



See ["Developing Context-Based Enrichments" on page 9](#) for more information.

Pivot-Based Enrichments

Pivot-based enrichments expand the breadth of data that analysts can use to build out graphical relationships during threat research on the Explore pivoting tool. Further, organizations that have enabled the latest Investigations user interface can leverage pivot-based enrichments on the pivoting tool within investigations. To enable the development of pivot-based enrichments, the Anomali Enrichments SDK includes the Anomali Transform Library.



See ["Developing Pivot-Based Enrichments" on page 23](#) for more information.

Adding enrichments to ThreatStream using the Enrichments SDK includes developing an enrichment bundle that adheres to the requirements in this guide and submitting it to Anomali for certification.

Developing Enrichments for ThreatStream Cloud

Adding enrichments to ThreatStream Cloud using the Enrichments SDK includes developing an enrichment bundle that adheres to the requirements in this guide and submitting it to Anomali for certification.

To create enrichments for use on ThreatStream Cloud:

1. Develop data enrichment scripts that Anomali will use to retrieve and display data from the external source. See ["Developing Pivot-Based Enrichments" on page 23](#) and ["Developing Context-Based Enrichments" on page 9](#).
2. Test your python scripts. See ["Testing Your Python Script" on page 29](#).
3. Create required enrichment metadata. See ["Creating Enrichment Metadata Files" on page 32](#).
4. Bundle your python script and enrichment metadata as a tarball. See ["Creating the Enrichment TGZ Bundle" on page 38](#).
5. Test your completed bundle on a ThreatStream staging server. See ["Testing ThreatStream Cloud Enrichment Bundles" on page 40](#).
6. Write customer facing documentation for your enrichment. See ["Providing Documentation for Your Enrichment" on page 46](#).
7. Submit the enrichment to Anomali. See ["Submitting ThreatStream Cloud Enrichments for Certification " on page 43](#).

Prerequisites

Before continuing, ensure that the following prerequisites are met:

- The python platform is on v3.7.x
- The python requests library is on v2.25.1
- The xmljson python library is on v0.2.1

The Enrichments SDK uses the following non-standard python libraries:

- requests
- dateutil

Therefore, you must run the following commands before locally testing enrichments:

- `pip install requests`
- `pip install python-dateutil`

Note: No other python packages should be installed on your local or svlpartner instance. All source code must be included in the source directory in a non-nested structure.

Chapter 2: Developing Context-Based Enrichments

In order to create enrichments that display enrichment data in widgets on observable details pages in ThreatStream, you must create a python script that utilizes the Anomali Enrichment Library. The python script must contain an enrichment object, which retrieves data from the enrichment source, and widget objects, which determine how enrichment data is displayed to users on observable details pages.

Understanding the Anomali Enrichment Library

This section contains information on creating enrichment objects using the Anomali Enrichment Library. The library—`anomalienrichment.py`—is available for reference and testing purposes in the Enrichments SDK package.

Method	Description	Return
<code>anomali_enrichment.parseArguments()</code>	Parses input arguments for the transform script.	None
<code>anomali_enrichment.getTransformName()</code>	Gets the name of the executed transform.	Transform Name (string)
<code>anomali_enrichment.getEntityValue()</code>	Gets the value of the entity on which the transform was executed.	Entity Value (String)
<code>anomali_enrichment.getFieldValue(String fieldName)</code>	Gets the value of additional field for the entity on which the transform was executed.	Field Value (String)
<code>anomali_enrichment.getCredentialValue(String credentialName)</code>	Gets required credential values for the data enrichment source.	Credential Value (String)

Method	Description	Return
<code>anomali_enrichment.addEntity(String entityType, String entityValue)</code>	Adds the entity to the transform object.	Entity Object
<code>anomali_enrichment.addMessage(String messageType, String messageText)</code>	Adds a type of message that will be displayed in the JSON output for debugging purposes. Possible values could include: "DEBUG", "INFO", "WARNING", "ERROR", "CRITICAL"). See "Creating Enrichment Bundles" on page 32 for information on viewing these messages.	None
<code>anomali_enrichment.addException(String exceptionString)</code>	Adds the text of the message that will be displayed on the ThreatStream user interface in case of errors.	None
<code>anomali_enrichment.returnOutput()</code>	Returns the result of the transform in JSON format.	Transform Result (JSON formatted String)

Widget Objects

This section contains information on the widgets provided by the Anomali Enrichment Library. Your enrichment can contain multiple widgets.

TextWidget Objects

TextWidget objects display data as formatted plain text. The following is an example of a TextWidget Object:

```
IP Enrichment for 104.27.158.93
```

TextWidget objects must be specified in python scripts as follows:

```
text_widget = TextWidget(ItemInWidget item, Boolean lineBreakEnding)
```

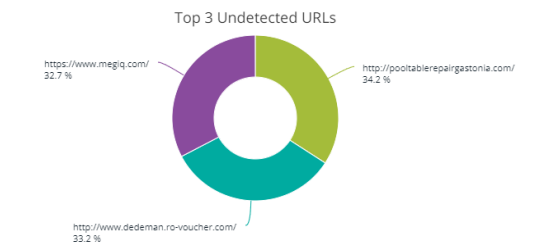
Method	Description	Return
<code>text_widget.setItem</code> (ItemInWidget item)	Sets the ItemInWidget object.	None
<code>text_widget.setLineBreakEnding</code> (Boolean lineBreakEnding)	Sets the boolean flag to indicate whether there is a line break following the text. Note: If the last widget in your script is a TextWidget, you must set this method to <code>true</code> .	None

See **6** under "Creating Your Context-Based Enrichment Script" on page 18 for a TextWidget example.

ChartWidget Objects

ChartWidget objects enable you to display data in a formatted chart.

The following is an example of a ChartWidget:



JSON strings developed with the Highcharts v6.2 library can be added to the enrichment. The Anomali Enrichments SDK supports any combination of the following Highcharts chart types:

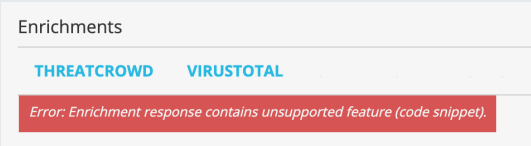
- area
- areaspline
- bar
- boxplot
- column
- columnrange
- errorbar

- gauge
- heatmap
- line
- pie
- spline
- waterfall
- xrange

Only the chart types listed above are supported. To read more about supported Highcharts chart types, see <https://www.highcharts.com/docs/chart-and-series-types/chart-types>

Notes:

- 3D chart options are not supported.
- Callback functions and properties such as allowHTML or useHTML are not supported. In these cases, enrichment responses are not rendered and the following error message is displayed:



Bundles containing such code snippets will be rejected during code review.

ChartWidget objects must be specified in python scripts as follows:

```
chart_widget = ChartWidget(String chartName, String highchartsJson)
```

Method	Description	Return
chart_widget.setChartName (String chartName)	Sets the name for the chart.	None
chart_widget.setHighchartsJson (String highchartsJson)	Sets the JSON string to be rendered by Highcharts in UI. The string you reference must be included in your python script. See 4 and 10 in "Creating Your Context-Based Enrichment Script" on page 18 for an example.	None

TableWidget Objects

TableWidget objects enable you to display data in a formatted table. When you add a TableWidget to an enrichment, the ThreatStream user interface automatically enables the sorting of data within columns.

The following is an example of a TableWidget:

25 ▾ 1 - 25 of 918 items

Hostname	Last Resolved
06903609.cn	2019-01-16 20:20:04
12fanclub.ml	2018-12-20 17:07:04
1387tr.com	2018-04-19 10:04:53
1742919.top	2016-11-17 00:00:00
2000k5.tw	2018-11-20 23:54:06
2grkgcnet.ml	2018-09-23 23:28:28
93115286.cn	2018-09-26 12:35:18
33d3d6.ga	2018-11-19 19:52:10
38mm.xyz	2019-01-24 13:23:15
3sootsport.com	2018-09-30 16:10:07
3tc.cloud	2019-02-09 22:12:04

TableWidget objects must be specified in python scripts as follows:

```
table_widget = TableWidget(String tableName, StringList columnHeadings,  
StringList columnTypes)
```

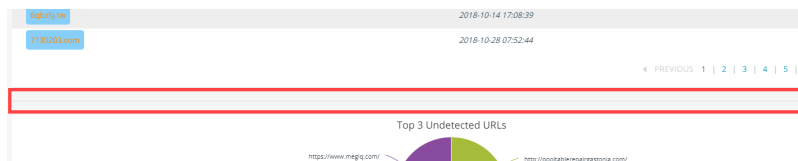
Method	Description	Return
<code>table_widget.setTableName (String tableName)</code>	Sets the name for the table.	None
<code>table_ widget.setColumnHeadings (StringList columnHeadings)</code>	Sets the table column headings as a list of strings.	None
<code>table_ widget.setColumnTypes (StringList columnTypes)</code>	(Optional) Sets the table column types as a list of strings(itemTypes). Note: If you do not specify a column type, each item in the table must be given an itemType.	None

Method	Description	Return
<code>table_widget.setColumnWidths</code> (StringList columnWidths)	Sets table column widths as a list of strings. For example: <code>columnWidths=['20%', '80%']</code> In the above example, column one is given a width of 20% and column two is given a width of 80%. If no width is specified, columns are rendered with equal width.	None
<code>table_widget.addRowOfItems</code> (ItemInWidgetList listOfItems)	Adds a row into the table as a list of ItemInWidget objects.	None

See **7** under "[Creating Your Context-Based Enrichment Script](#)" on page 18 for a TableWidget example.

HorizontalRuleWidget Objects

HorizontalRuleWidget objects enable you to add lines between widgets in your enrichment. The following is an example of a HorizontalRuleWidget:



HorizontalRuleWidget objects must be specified in python scripts as follows:

```
horizontalrule_widget = HorizontalRuleWidget()
```

See **8** under "[Creating Your Context-Based Enrichment Script](#)" on page 18 for a HorizontalRuleWidget example.

ItemInWidget Objects

ItemInWidget objects are used to format content within widgets, such as table cells, or text of any kind.

ItemInWidget objects must be specified in python scripts as follows:

```
item_in_widget = ItemInWidget(String itemType, String itemValue, String  
itemLabel, String backgroundColor, String textColor, String fontSize,  
String fontStyleWeight)
```

Method	Description	Return
<code>item_in_widget.setType</code> (String itemType)	<p>Sets the type for the item. Available types include:</p> <ul style="list-style-type: none">• String = "String"• Integer = "Integer"• Float = "Float"• Date = "Date"• DateTime = "DateTime"• Link = "Link"• Domain = "Domain"• IPv4 = "IPv4"• Hash = "Hash"• Email = "Email"• URL = "URL"• Phrase = "Phrase"• AS = "AS"• NSRecord = "NSRecord" <p>Note: Use the Link type for any URL you want to make clickable. All other types render the item as text.</p> <p>The following is an example of a valid link:</p> <pre>anomali_enrichment.addWidget (TextWidget(ItemInWidget (ItemTypes.Link, "https://www.example.com" % search_string, "Click here")),</pre>	None

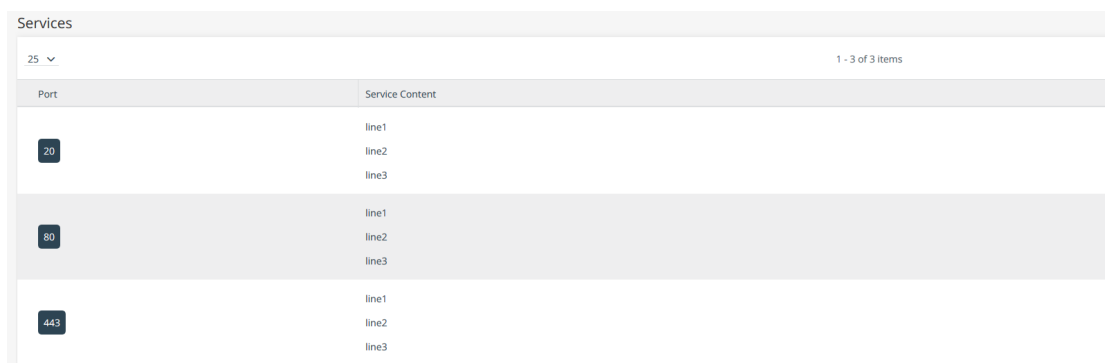
Method	Description	Return
	False))	
item_in_widget.setValue (String itemValue)	Sets the value for the item.	None
item_in_widget.setLabel (String itemLabel)	Sets the label for the item.	None
item_in_widget.setBackgroundColor (String backgroundColor)	<p>(Optional) Sets the background color for the item.</p> <p>If you do not specify a value, ThreatStream uses a default color.</p> <p>Tip: You can use any of the 140 standard color names supported by HTML, such as <i>SteelBlue</i>. You can also specify a hexadecimal value, such as <i>#4682B4</i>.</p>	None
item_in_widget.setTextColor(String textColor)	<p>(Optional) Sets the text color for the item.</p> <p>If you do not specify a value, ThreatStream uses a default color.</p> <p>Tip: You can use any of the 140 standard color names supported by HTML, such as <i>SteelBlue</i>. You can also specify a hexadecimal value, such as <i>#4682B4</i>.</p>	None

Method	Description	Return
<code>item_in_widget.setFontSize</code> (String fontSize)	(Optional) Sets the font size for the item. If you do not specify a value, ThreatStream uses a default font size. Tip: Valid values include <code>xx-small</code> , <code>x-small</code> , <code>small</code> , <code>medium</code> , <code>large</code> , <code>x-large</code> , and <code>xx-large</code> . You can also specify a pixel size, such as <code>16px</code> .	None
<code>item_in_widget.setFontStyleWeight</code> (String fontStyleWeight)	(Optional) Sets the font style and weight for the item. If you do not specify a value, ThreatStream uses a default font style. Tip: Valid values include <code>normal</code> , <code>italic</code> , <code>bold</code> , and <code>italic-bold</code> .	None

Compositeltem Objects

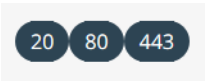
Compositeltem Objects enable you to display multiple Text Widgets in a single line or create multiple lines within a Table Widget cell.

The following is an example of a Compositeltem within a Table Widget:



Services	
25 ▾	1 - 3 of 3 items
Port	Service Content
20	line1 line2 line3
80	line1 line2 line3
443	line1 line2 line3

The following is an example of a Compositeltem within a Text Widget:



CompositeItem Objects must be specified in python scripts as follows:

```
composite_item = CompositeItem(Boolean onSeparateLines)
```

Method	Description	Return
composite_item.addItemInWidget (ItemInWidget itemInWidget)	Adds an ItemInWidget object to the CompositeItem object.	ItemInWidget Object
composite_item.setOnSeparateLines (Boolean onSeparateLines)	Sets the boolean flag to indicate whether objects are displayed on separate lines.	None

See [10](#) and [11](#) under "[Creating Your Context-Based Enrichment Script](#)" below for CompositeItem examples.

Creating Your Context-Based Enrichment Script

Depending on the data source and its specific requirements, the content of enrichment python scripts will vary. The example described below is a truncated version of the example context-based enrichment script—vt_anomali_enrichment.py—included in the Anomali Enrichments SDK package. The script includes three enrichments and provides a high level illustration of what is required for enrichments to function on observable details pages.

```
import os
import sys
import copy
import json
import requests

from AnomaliEnrichment import AnomaliEnrichment, TextWidget, ChartWidget, TableWidget, HorizontalRuleWidget, \
    ItemInWidget, ItemTypes, CompositeItem

api_base = "https://www.virustotal.com/vtapi/v2/"
api_key = None

piechart_template_dict = {
    "chart": {
        "height": 300,
        "margin": 0,
        "marginTop": -20,
        "plotBackgroundColor": None,
        "plotBorderWidth": None,
        "backgroundColor": "#fff"
    },
    "credits": {
        "enabled": False
    },
    "title": {
        "text": "Pie Chart Example"
    },
    "tooltip": {
        "headerFormat": "<span style='font-size: 10px'>{point.key}</span><br>",
        "pointFormat": "<b>{point.y} ({point.percentage:.1f}%)</b>"
    },
    "plotOptions": {
        "pie": {
            "allowPointSelect": True,
            "animation": True,
            "cursor": "pointer",
            "innerSize": 100,
            "dataLabels": {
                "enabled": True,
                "format": "{point.prettyName}<br>{point.percentage:.2f} %",
                "style": {
                    "width": "200px"
                }
            },
            "point": {
                "events": {}
            },
            "size": "45%",
            "center": ["50%", "60%"]
        }
    },
    "series": [
        {
            "type": "pie",
            "name": "",
            "data": []
        }
    ]
}
```

- ❶ Imports the system level libraries needed to execute your script.
- ❷ Imports the Anomali Enrichment Library.
- ❸ Specifies the API of the external data source and any necessary credentials.
- ❹ Highcharts JSON template to be referenced by the chart widget.

```
def enrichDomainTabOne(anomali_enrichment, search_string):  
    try:  
        response = requests.get(api_base + 'domain/report?apikey=' + api_key + '&domain=' + search_string)  
        response_json = response.json()  
        resp_code = int(response_json['response_code'])  
        if resp_code == 1:  
            anomali_enrichment.addWidget(TextWidget(ItemInWidget(ItemTypes.String,  
                                                                "Domain Enrichment for %s" % search_string,  
                                                                "Domain Enrichment for %s" % search_string,  
                                                                "SteelBlue", "White", "large", "bold"), True))  
            anomali_enrichment.addWidget(TextWidget(ItemInWidget(ItemTypes.Link,  
                                                                "https://www.virustotal.com/#/domain/%s" % search_string,  
                                                                "View"), False))  
            anomali_enrichment.addWidget(TextWidget(ItemInWidget(ItemTypes.String,  
                                                                "comparison"), True))  
            if 'resolutions' in response_json:  
                table_widget = TableWidget("Resolutions", [{"IP Address", "Last Resolved", "Reference"}])  
                for resolution in response_json['resolutions']:  
                    table_widget.addRowOfItems([ItemInWidget(ItemTypes.IPv4,  
                                                            resolution['ip_address'],  
                                                            resolution['ip_address']),  
                                                ItemInWidget(ItemTypes.DateTime,  
                                                            resolution['last_resolved'],  
                                                            resolution['last_resolved']),  
                                                ItemInWidget(ItemTypes.Link,  
                                                            "https://www.virustotal.com/#/ip-address/%s" % resolution['ip_address'],  
                                                            "detail")])  
            anomali_enrichment.addWidget(table_widget)  
        # Create HorizontalRuleWidget  
        anomali_enrichment.addWidget(HorizontalRuleWidget())
```

- 5 Enrichment 1:** Retrieves data from the external source.
- 6 Enrichment 1:** Adds a text widget to the enrichment. See ["TextWidget Objects"](#) on page 10 for an example of this widget.
- 7 Enrichment 1:** Adds a table widget to the enrichment. See ["TableWidget Objects"](#) on page 13 for an example of this widget.
- 8 Enrichment 1:** Adds a horizontal rule widget to the enrichment. See ["HorizontalRuleWidget Objects"](#) on page 14 for an example of this widget.

```
def create_chart_widget(template_dict, data):
    graph_dict = copy.deepcopy(template_dict)
    # fill in the data
    graph_dict['series'][0]['data'] = data
    graph_json = json.dumps(graph_dict)
    pie_chart_widget = ChartWidget("Requester Distribution", graph_json)
    return pie_chart_widget

data = [{'y': 0.8, 'prettyName': u'US', 'name': u'US'},
        {'y': 0.1, 'prettyName': u'DE', 'name': u'DE'},
        {'y': 0.1, 'prettyName': u'TH', 'name': u'TH'}]
# Create Pie ChartWidget
chart_widget = create_chart_widget(piechart_template_dict, data)
anomaly_enrichment.addWidget(chart_widget)

# Create composite_item for TextWidget
# always declare a new CompositeItem before using it
text_composite_item = CompositeItem(onSeparateLines=False)
port_list = [20, 80, 443]
for port in port_list:
    text_composite_item.addItemInWidget(ItemInWidget(ItemTypes.String, str(port), textColor='#ffffff',
                                                       backgroundColor='#2D4453', fontSize='medium'))
port_text_widget = TextWidget(text_composite_item, True)
anomaly_enrichment.addWidget(port_text_widget)

# Create composite_item for TableWidget
services_table_widget = TableWidget(tableName='Services', columnHeadings=['Port', 'Service Content'],
                                     columnTypes=[ItemTypes.Integer, ItemTypes.String],
                                     columnWidths=['20%', '80%'])

# In real world, get data from the API response
services_data = [{'port': 20, 'lines': ['line1', 'line2', 'line3']},
                  {'port': 80, 'lines': ['line1', 'line2', 'line3']},
                  {'port': 443, 'lines': ['line1', 'line2', 'line3']}]

for service_data in services_data:
    port = service_data['port']
    # always declare a new CompositeItem before using it
    table_composite_item = CompositeItem(onSeparateLines=True)
    for line in service_data['lines']:
        table_composite_item.addItemInWidget(ItemInWidget(ItemTypes.String, line))
    services_table_widget.addRowOfItems([ItemInWidget(ItemTypes.String, port,
                                                       textColor='#ffffff', backgroundColor='#2D4453'),
                                         table_composite_item])
anomaly_enrichment.addWidget(services_table_widget)

except:
    anomaly_enrichment.addException('enrichDomain Unknown Error: %sType: %sValue: %s' %
                                    (os.linesep, sys.exc_info()[0], os.linesep, sys.exc_info()[1]))
return anomaly_enrichment
```

9 Enrichment 1: Adds a chart widget to the enrichment. The code in this section gathers and injects data into the chart template defined in **4**. See ["ChartWidget Objects" on page 11](#) for an example of this widget.

10 Enrichment 1: Adds a text widget that contains composite item objects. See ["CompositeItem Objects" on page 17](#) for an example of this object.

11 Enrichment 1: Adds a table widget that contains composite item objects. See ["CompositeItem Objects" on page 17](#) for an example of this object.

11 Enrichment 1: Specifies the exception—what should be done in cases where no data is returned.

```
def enrichDomainTabTwo(anomali_enrichment, search_string):  
    (... omitted for brevity ...)  
  
def enrichIP(anomali_enrichment, search_string):  
    (... omitted for brevity ...)  
  
functions = {  
    'enrichDomainTabOne': enrichDomainTabOne,  
    'enrichDomainTabTwo': enrichDomainTabTwo,  
    'enrichIP': enrichIP  
}  
  
if __name__ == '__main__':  
    anomali_enrichment = AnomaliEnrichment()  
    anomali_enrichment.parseArguments()  
    transform_name = anomali_enrichment.getTransformName()  
    entity_value = anomali_enrichment.getEntityValue()  
    api_key = anomali_enrichment.getCredentialValue('api_key')  
  
    functions[transform_name](anomali_enrichment, entity_value)  
    anomali_enrichment.ReturnOutput()
```

- 13 Enrichment 2:** Omitted for brevity. To view this enrichment code, see the example script in the Anomali Enrichments SDK package.
- 14 Enrichment 3:** Omitted for brevity. To view this enrichment code, see the example script in the Anomali Enrichments SDK package.
- 15** Specifies the list of enrichments supported by the script.
- 16** Initializes the Anomali Enrichment SDK object and input parameters.
- 17** Retrieves the credentials for your data enrichment source from STDIN using Anomali Enrichment SDK object.
- 18** Returns data enrichment output to STDOUT in Anomali Enrichment format.

Chapter 3: Developing Pivot-Based Enrichments

In order to create data enrichments that can be leveraged on the standalone Explore pivoting tool and the pivoting tool on the Investigations user interface within ThreatStream, you must create transforms that utilize the Anomali Transform Library. ThreatStream will use these transforms to dynamically enrich threat intelligence with data from the external enrichment source.

Note: Anomali Enrichments SDK also supports Maltego transforms for purposes of backwards compatibility. This guide does not include information on creating ThreatStream enrichments that leverage Maltego Transforms.

Understanding the Anomali Transform Library

Transforms must be constructed using the Anomali Transform Library. The library—`anomalitransform.py`—is available for reference and testing purposes in the Enrichments SDK package.

Transform Objects

This section contains information on creating transform objects using the Anomali Transform Library.

Method	Description	Return
<code>at.parseArguments()</code>	Parses input arguments for the transform script.	None
<code>at.getTransformName()</code>	Gets the name of the executed transform.	Transform Name (string)
<code>at.getEntityValue()</code>	Gets the value of the entity on which the transform was executed.	Entity Value (String)

Method	Description	Return
<code>at.getFieldValue(String fieldName)</code>	Gets the value of additional field for the entity on which the transform was executed.	Field Value (String)
<code>at.getCredentialValue(String credentialName)</code>	Gets required credential values for the data enrichment source.	Credential Value (String)
<code>at.addEntity(String entityType, String entityValue)</code>	Adds the entity to the transform object.	Entity Object
<code>at.addMessage(String messageType, String messageText)</code>	Adds a type of message that will be displayed in the JSON output for debugging purposes. Possible values could include: "DEBUG", "INFO", "WARNING", "ERROR", "CRITICAL"). See "Creating Enrichment Bundles" on page 32 for information on viewing these messages.	None
<code>at.addException(String exceptionString)</code>	Adds the text of the message that will be displayed on the ThreatStream user interface in case of errors.	None
<code>at.returnOutput()</code>	Returns the result of the transform in JSON format.	Transform Result (JSON formatted String)

Entity Objects

This section contains information on specifying entity objects using the Anomali Transform Library.

The following methods must be used when specifying entity objects:

Method	Description	Return
<code>ae.setType(String entityType)</code>	<p>Sets the type for the entity. Available types include:</p> <ul style="list-style-type: none">• Domains Example: <code>EntityTypes.Domain = "anomali.Domain"</code>• IP addresses Example: <code>EntityTypes.Ipv4 = "anomali.Ipv4Address"</code>• Hashes Example: <code>EntityTypes.Hash = "anomali.Hash"</code>• Email addresses Example: <code>EntityTypes.EmailAddress = "anomali.EmailAddress"</code>• URLs Example: <code>EntityTypes.URL = "anomali.URL"</code>• Phrases Example: <code>EntityTypes.Phrase = "anomali.Phrase"</code>• Autonomous system numbers Example: <code>EntityTypes.AS = "anomali.AS"</code>• DNS name server records	None

Method	Description	Return
	Example: EntityTypes.NSRecord = "anomali.NSRecord"	
ae.setValue(String entityValue)	Set the value for the entity.	Transform Name (string)
ae.addAdditionalField(String fieldName, String displayName, String fieldValue)	Add an additional field to the entity.	Entity Value (String)

Creating Your Pivot-Based Enrichment Script

Depending on the data source and its specific requirements, the content of enrichment python scripts will vary. The example described below is identical to the example pivot-based script—`vt_anomali_transform.py`— provided by the Anomali Enrichments SDK package. The example below provides a high level illustration of what is required for pivoting enrichments to function.

```
import os
import sys
import requests

from AnomaliTransform import AnomaliTransform
from AnomaliTransform import EntityTypes

api_base = "https://www.virustotal.com/vtapi/v2/"
api_key = None

def domainToIP(at, search_string):
    try:
        response = requests.get(api_base + 'domain/report?apikey=' + api_key + '&domain=' + search_string)
        response_json = response.json()
        resp_code = int(response_json['response_code'])
        if resp_code == 1:
            if 'resolutions' in response_json:
                for resolutions in response_json['resolutions']:
                    ae = at.addEntity(EntityTypes.IPv4, '%s' % resolutions['ip_address'])
                    ae.addAdditionalField('last_resolved', 'Last Resolved', '%s' % resolutions['last_resolved'])
            else:
                at.addException('domainToIP Unknown Error:%sType: %s%sValue:%s' %
                                (os.linesep, sys.exc_info()[0], os.linesep, sys.exc_info()[1]))
        return at

functions = {
    'domainToIP': domainToIP,
}

if __name__ == '__main__':
    at = AnomaliTransform()
    at.parseArguments()
    transform_name = at.getTransformName()
    entity_value = at.getEntityValue()
    api_key = at.getCredentialValue('api_key')

    functions[transform_name](at, entity_value)
    at.returnOutput()
```

- 1 Imports the system level libraries needed to execute your script.
- 2 Imports the Anomali Transform Library.
- 3 Specifies the API of the external data source and any necessary credentials.
- 4 Section of the transform that retrieves data from the external source.
- 5 Section of the transform that formats the data for consumption by Anomali.
- 6 Specifies the exception—what should be done in cases where no data is returned.
- 7 Specifies the list of transforms supported by the script.
- 8 Initializes the Anomali Transform SDK object and input parameters.
- 9 Retrieves the credentials for your data enrichment source from STDIN using Anomali Transform SDK object.
- 10 Returns data enrichment output to STDOUT in Anomali Transform format.

Chapter 4: Testing Your Python Script

When development on a python script is complete, you can use the instructions in this section to test it on your local machine.

To test your script locally:

1. Create a virtual environment:

```
virtualenv venv
```

2. Activate the virtual environment:

```
source venv/bin/activate
```

3. Install the requests library.

```
(venv) pip install requests
```

4. Use your preferred method to copy the script you want to test and the `anomalienrichment.py` (for context-based enrichments) or `theanomalitransform.py` (for pivot-based enrichments) file to the same directory.

5. Execute a transform or enrichment contained in the script:

```
venv) python <script_name>.py <transform_or_enrichment_name> <entity value>
--credentials "{\"api_key\":\"<api_key_value>\"}"
```

For example, to test a transform in a pivot-based script, you would run:

```
(venv) python vt_anomali_transform.py domainToIP anomali.com --
credentials "{\"api_key\":\"REDACTED\"}"
```

To test an enrichment in a context-based script, you would run:

```
(venv) python vt_anomali_enrichment.py enrichDomain anomali.com --
credentials "{\"api_key\":\"REDACTED\"}"
```

For context-based enrichments, you will see output similar to the following:

```
"exceptions": [],
"messages": [],
"widgets": [
  {
    "widgetType": "Text",
    "item": {
      "backgroundColor": "SteelBlue",
      "fontSize": "large",
      "fontStyleWeight": "bold",
      "itemLabel": "Domain Enrichment for anomali.com",
      "itemType": "String",
      "itemValue": "Domain Enrichment for anomali.com",
      "textColor": "White"
    },
    "lineBreakEnding": true
  },
  {
    "widgetType": "Table",
    "tableName": "Resolutions",
    "columnHeadings": [
      "IP Address",
      "Last Resolved"
    ],
    "rows": [
      [
        {
          "itemLabel": "216.218.192.90",
          "itemType": "IPv4",
          "itemValue": "216.218.192.90"
        },
        {
          "itemLabel": "2019-02-07 19:05:40",
          "itemType": "DateTime",
          "itemValue": "2019-02-07 19:05:40"
        }
      ],
      [
        {
          "itemLabel": "64.62.160.174",
          "itemType": "IPv4",
          "itemValue": "64.62.160.174"
        },
        {
          "itemLabel": "2015-12-06 00:00:00",
          "itemType": "DateTime",
          "itemValue": "2015-12-06 00:00:00"
        }
      ]
    ]
  },
  {
    "widgetType": "HorizontalRule"
  },
  {
    "widgetType": "Chart",
    "chartName": "Top 3 Undetected Referrer Samples",
    "highchartsUson":
      "({\"chart\":{\"height\":300,\"margin\":0,\"marginTop\":-20,\"plotBackgroundColor\":null,\"plotBorderWidth\":null,\"background\":\"#fff\"},\"credits\":{\"enabled\":false},\"title\":{\"text\":\"Top 3 Undetected Referrer Samples\"},\"tooltip\":{\"headerFormat\":\"<span style=\\\"font-size: 10px\\\">{point.key}</span><br/>\", \"pointFormat\":\"<b>{point.y}</b>({point.percentage:.1f})</b>\", \"plotOptions\":{\"pie\":{\"allowPointSelect\":true,\"animation\":false,\"cursor\":\"pointer\", \"innerSize\":100,\"dataLabels\":{\"enabled\":true,\"format\":\"{point.prettyName}<br>{point.percentage:.1f}\"}, \"style\":{\"width\":\"200px\"}}, \"point\":{\"events\":{}}, \"size\":\"65%\"}}, \"series\":[{ \"type\":\"pie\", \"name\":\"\", \"data\":{\"y\":71,\"prettyName\":\"807ed3a83aeb8731502639e50726aada10433fef3eb306f92ee54d13c2a5a344\", \"name\":\"807ed3a83aeb8731502639e50726aada10433fef3eb306f92ee54d13c2a5a344\"}, { \"y\":70,\"prettyName\":\"82c989f1871d627a9a3bedda0ccf9a113c0c61491ee7a638b50955355382dd7\", \"name\":\"82c989f1871d627a9a3bedda0ccf9a113c0c61491ee7a638b50955355382dd7\"}, { \"y\":70,\"prettyName\":\"77fcd2ad32felbe4b7bdd272faf95ab8e18404f1967ded44a9db730171691ab\", \"name\":\"77fcd2ad32felbe4b7bdd272faf95ab8e18404f1967ded44a9db730171691ab\"}}]})"
```

For pivot-based enrichments, you will see output similar to the following:

```
{
  "entities": [
    {
      "additionalFields": [
        {
          "displayName": "Last Resolved",
          "fieldName": "last_resolved",
          "fieldValue": "2018-09-04 03:17:59"
        }
      ],
      "entityType": "anomaly.IPv4Address",
      "entityValue": "216.218.192.90"
    },
    {
      "additionalFields": [
        {
          "displayName": "Last Resolved",
          "fieldName": "last_resolved",
          "fieldValue": "2015-12-06 00:00:00"
        }
      ],
      "entityType": "anomaly.IPv4Address",
      "entityValue": "64.62.160.174"
    }
  ],
  "exceptions": [],
  "messages": []
}
```

As these example show, the output must be in valid JSON format. The output must not contain messages that result from print statements. To add messages for debugging purposes, use the `at.addMessage` method, as described in ["Testing Your Python Script" on page 29](#).

Notes:

- Enrichment transforms that take longer than 30 seconds to complete will result in a timeout on the ThreatStream UI.
- Virtual environments created for testing your script must not be included in the final enrichment bundle.

Chapter 5: Creating Enrichment Bundles

Once you have developed and tested your enrichment python scripts, you must create a bundle that contains your python scripts and other supporting files. The bundle must contain the following items:

1. Python enrichment scripts

The enrichment scripts you developed must be included in the bundle within a directory called `source`.

2. Metadata file in JSON format

Metadata files must be created using the guidelines in ["Creating Enrichment Metadata Files" below](#). The metadata file must be named `metadata.json`

3. Enrichment Icons

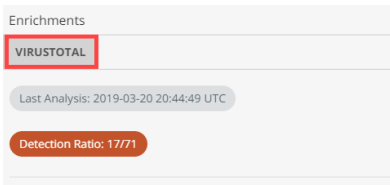
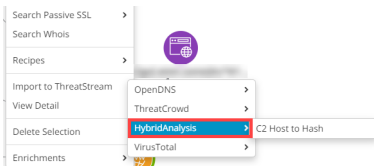
You must provide a full size icon for your enrichment, in addition to an icon thumbnail.

Creating Enrichment Metadata Files

In order to display enrichments on the ThreatStream user interface, you must create a JSON file that includes information such as enrichment names, transform names, icons, required credentials, and so on.

Items that must be specified in the metadata file include:

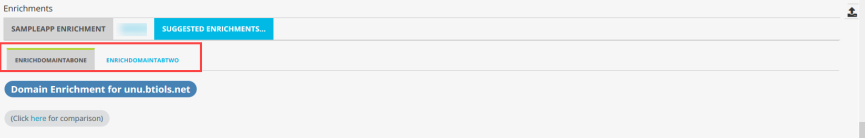
Metadata Item	Description
<code>title</code>	Name of the data enrichment source.
<code>is_python3</code>	If using the Anomali Enrichments SDK v3.0, you must set <code>is_python3</code> to <code>true</code> .

Metadata Item	Description
transform_set_name	<p>Name of the enrichment to be displayed on the ThreatStream user interface.</p> <p>For context-based enrichments, this sets the name of the tab in the Enrichments section on observable details pages.</p>  <p>For pivot-based enrichments, this sets the name of the menu item on the Explore pivoting tool which contains your transforms.</p>  <p>Displayed on the Explore user interface during pivoting.</p> <p>Note: You can only set one transform_set_name per bundle.</p> <p><i>For ThreatStream OnPrem and AirGap enrichments only:</i> you can use transform_set_name to distinguish enrichments undergoing testing from finalized enrichments.</p> <p>For example: sampleapp-anomali-beta</p>
description	Description of the data provided by the source. Displayed on the Enrichments tab within ThreatStream settings.
icon_display	Filename of the icon that will be displayed on the ThreatStream user interface.
icon_thumbnail	Filename of the icon thumbnail that will be displayed on the ThreatStream user interface.
sdk_type	Type of transform—you must specify <code>anomali</code> .

Metadata Item	Description
app_name	<p>Unique identifier used by ThreatStream to track enrichments. Must follow the format <domain_suffix>.<company_name>.<app-name></p> <p>For example: com.anomali.sampleapp</p> <p>If updates to your enrichment are ever required, simply ensure an identical app_name is used and re-upload your updated bundle.</p> <p><i>For ThreatStream OnPrem and AirGap enrichments only:</i> you can use the app_name to distinguish enrichments undergoing testing from finalized enrichments.</p> <p>For example: com.anomali.sampleapp-beta</p>
version	<p>Version of the enrichment. Must follow the format <major_version>.<minor_version>.<patch></p> <p>For example: 3.0.1</p> <p><i>For ThreatStream OnPrem enrichments only:</i> you can use the version number to distinguish enrichments undergoing testing from finalized enrichments.</p> <p>For example: 1.0.0-beta</p>
update_creds	<p>Set this parameter to true if you are adding a new credential field during an update to your enrichment. See "Updating Enrichment Credentials" on page 47 for more information.</p>

Metadata Item	Description
credentials	<p>Specifies credential requirements for users activating the enrichment. Items include:</p> <ul style="list-style-type: none">• <code>name</code>—name used to identify the credential field in your python script. Not displayed on the ThreatStream user interface. <p>Example: <code>api_key</code></p> <ul style="list-style-type: none">• <code>description</code>—description of the credential. This description is for internal purposes and not displayed on the ThreatStream user interface.• <code>label</code>—name of credential displayed to users on the ThreatStream user interface. <p>Example: API Key</p> <div>Note: Unicode characters are not supported.</div> <ul style="list-style-type: none">• <code>required</code>—whether users are required to enter the credential for enrichment activation. Specify <code>true</code> or <code>false</code>.• <code>sensitive</code>—whether the credential is case-sensitive. Specify <code>true</code> or <code>false</code>.• <code>rank</code>—order in which credential fields appear on the user interface. <p>Example: <code>"rank": 1</code> (appears at the top of the list), <code>"rank": 2</code> (appears second in the list)</p> <div>Note: Each credential field must be given a ranking if specifying rank for your credential fields. Rank will not be honored if you specify rank for only a partial set of credential fields.</div>

Metadata Item	Description
transforms	<p>List of features supported by the data enrichment script. You must specify values for:</p> <ul style="list-style-type: none"> • <code>transform_name</code>—name used to identify the transform in your python script. • <code>pivoting</code>—whether the transform is leveraged on the Explore pivoting tool. • <code>enrichment</code>—whether the transform is leveraged on observable details pages. <p>Note: For a given transform, you must set either <code>pivoting</code> <i>OR</i> <code>enrichment</code> to <code>true</code>. You <i>cannot</i> set both <code>pivoting</code> and <code>enrichment</code> to <code>true</code>. A single transform cannot be used on both context-based and pivot-based enrichments.</p> <p>Optional transform metadata items include:</p> <ul style="list-style-type: none"> • <code>activation</code>—set to <code>true</code> if you want Anomali to use the transform for validation when evaluating your enrichment. If you set <code>activation</code> to <code>true</code>, you must specify a <code>sample_entity_value</code>. • <code>sample_entity_value</code>—example observable value on which the transform will return data. Anomali will execute the transform on this value to validate your enrichment. <p>Multi-Transform Context Based Enrichments</p> <p>If your metadata file contains multiple context-based transforms (<code>"enrichment": true</code>) for the same <code>entity_type</code>, each transform is assigned a dedicated tab in your enrichment. A maximum of 10 transforms (and thus 10 tabs) are allowed.</p> <p>Only the transform on the first tab is executed when users access your multi-transform context based enrichment, thus reducing the chance of timeouts. Transforms on subsequent tabs are not executed until users open the respective tabs.</p> <p>Tabs are ordered based on the <code>rank</code> of the transforms for each</p>

Metadata Item	Description
	<p>entity_type (indicator type) in the metadata file. Values specified for transform_name are used for tab labels.</p> <p>The following is an example of a multi-tab enrichment:</p>  <p>Note: Tabs must load within 30 seconds to prevent timeouts.</p> <ul style="list-style-type: none"> display_name—name of the transform displayed to users on the ThreatStream user interface. description—description of the transform. author—the name of your organization. parameters— filename of the python script which contains the transform and the name of the transform. <p>Example: sample_enrichment.py ipToDomain</p> <ul style="list-style-type: none"> entity_type—type of entity on which the transform can be activated in ThreatStream. rank—order in which transforms appear on the user interface for the specified entity_type (indicator type). <p>Example: "rank": 1 (appears at the top of the list), "rank": 2 (appears second in the list)</p> <p>Note: Each transform must be given a ranking if specifying rank for your transforms. Rank will not be honored if you specify rank for only a partial set of available transforms.</p>
author	(Optional) Developer of the enrichment.

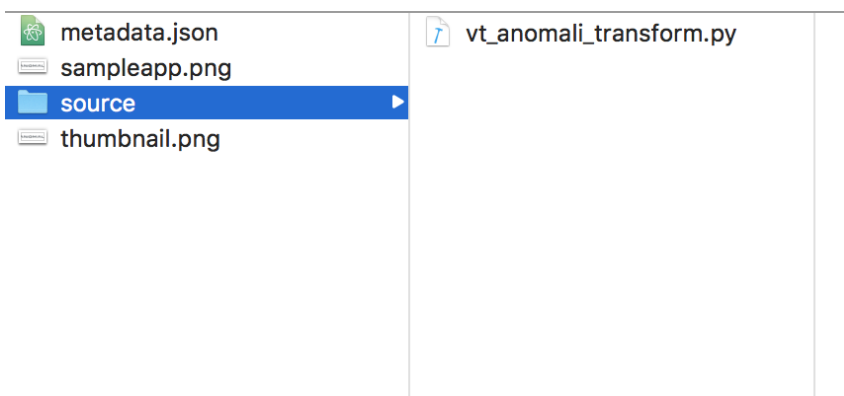
Metadata Item	Description
long_description	(Optional) Detailed description of the enrichment. Not currently displayed on the ThreatStream user interface.
license	(Optional) License for the enrichment scripts.
source_url	(Optional) URL for where the enrichment scripts are hosted.

An example metadata file is included in the Enrichments SDK package provided by Anomali.

Creating the Enrichment TGZ Bundle

Once you have assembled the python scripts, metadata file, and icon images, use the instructions in this section to create the final TGZ bundle for the enrichment.

The following is an example of a directory that contains all required files for the Enrichment SDK bundle:



To create the enrichment bundle:

1. Create a directory at the same level as your enrichment folder:

```
mkdir bundle
```

2. Move to the directory that contains the enrichment bundle components:

```
cd <dir_name>
```

3. Bundle the enrichments components in TGZ format:

```
tar cvf ../bundle/<file_name>.tgz .
```

Note: MacOS users must use the `COPYFILE_DISABLE` flag for the bundle to be compatible with ThreatStream. As such MacOS users should use the following command:

```
COPYFILE_DISABLE=1 tar cvf ../bundle/<file_name>.tgz .
```

If you intend to use your enrichment on ThreatStream Cloud, proceed to ["Testing ThreatStream Cloud Enrichment Bundles"](#) on page 40.

Chapter 6: Testing ThreatStream Cloud Enrichment Bundles

If you are developing your enrichment for use on ThreatStream Cloud, you must test your enrichment on a staging server before submitting it to Anomali for certification.


To gain access to a staging instance, send an email to ***enrichments.sdk@anomali.com*** that includes the following information:

- Organization name
- Email addresses of all users in need of access
- Which users should be given administrative privileges
- IP addresses from which you will access the staging server

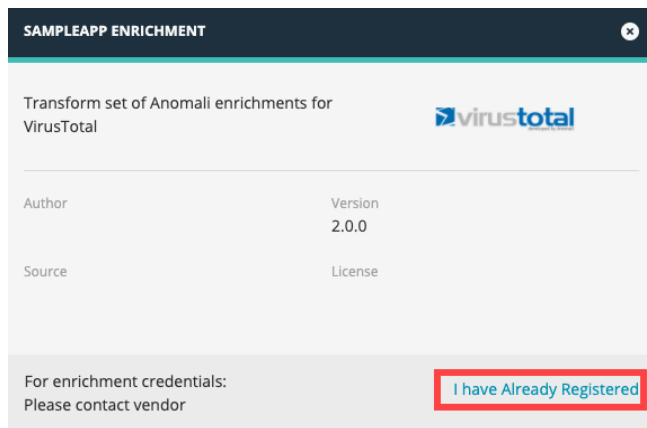
After your request is approved, Anomali will email you with further instructions on accessing the staging instance. Once you have access, you can connect to the staging instance and upload the bundle for testing.

Note: *Do not add users or change user privileges through the staging server user interface.* Send all staging server user administration requests to ***enrichments.sdk@anomali.com***.

To upload and activate your enrichment:

1. Connect to the ThreatStream staging user interface.
2. In the top navigation bar, click  and then **Integrations**.
3. Click **Upload New Enrichment** and browse for the TGZ file on your system.
4. Click **Upload & Install**. After installation, the enrichment is displayed on the Integrations screen.
5. Locate your enrichment and click **Set Up**.

6. If credentials are required, click **I have already registered** and enter the credentials for your enrichment.



SAMPLEAPP ENRICHMENT

Transform set of Anomali enrichments for VirusTotal

Author	Version
	2.0.0
Source	License

For enrichment credentials:
Please contact vendor

I have Already Registered

7. Click **Activate**.

Your enrichment is now active on ThreatStream staging and ready to test.

Note: Only users with Org Admin privileges can upload and install enrichments.

Debugging Enrichments

If you added any debugging messages to your enrichment using the `at.addMessage` method, they will be displayed in the JSON output of the enrichment.

To view debugging messages:

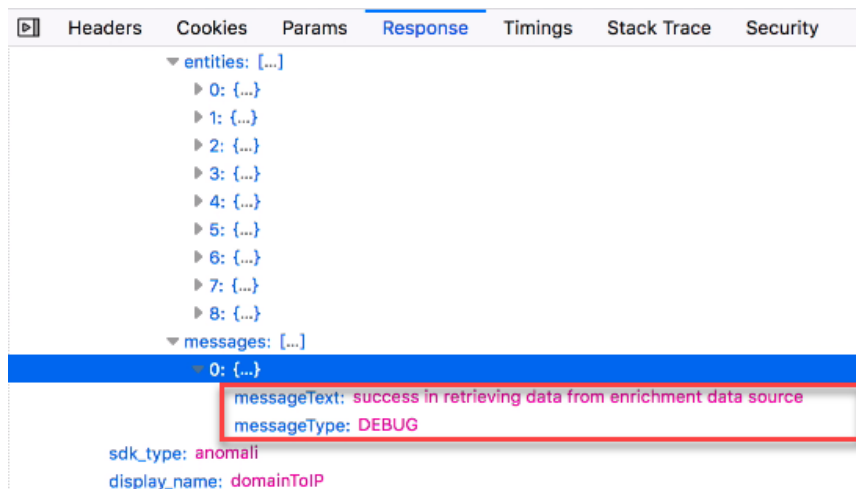
1. Open the developer console on your web browser.
2. To debug a context-based enrichment, navigate to the details page of an observable on which enrichment data is returned and open the tab for the enrichment under Enrichments.

To debug a pivot-based enrichment, navigate to **Research > Explore**, add an observable to the chart, and run the enrichment you want to debug.

3. If using Google Chrome, open the Network tab of the developer console.

If using another browser, locate a comparable resource.

4. Use the developer console search to locate a POST request that uses `https://svlpartner-optic-api.threatstream.com/api/v1/integration_package/transform/`.
5. Debugging messages are displayed in the `messages` field under `transform_result`.



Chapter 7: Submitting ThreatStream Cloud Enrichments for Certification

Once you have tested your enrichment intended for use on ThreatStream Cloud you can submit it to Anomali for certification. To submit your enrichment, include the following in an email to enrichments.sdk@anomali.com:

✓	Requirement	Description
Vendor and Enrichment Information		
✓	Vendor Name	Name of the enrichment developer organization.
✓	Enrichment Name	Name of the enrichment to be displayed on the ThreatStream user interface. <i>Maximum length: 100 characters</i> <i>Recommended length: 40 or fewer characters</i> Provide the same value you specified for <code>title</code> in the metadata file.
✓	Enrichment Description	Description of the enrichment to be displayed to users on the ThreatStream user interface. <i>Maximum length: 255 characters including spaces</i> Provide the same value you specified for <code>description</code> in the metadata file.
✓	Enrichment Version	Version number of the enrichment. Provide the same value you specified for <code>version</code> in the metadata file.

✓	Requirement	Description
✓	Enriched Entities	Types of data on which the enrichment can be executed. Include example values on which Anomali can test the enrichment. <i>Examples:</i> <ul style="list-style-type: none"> • <i>Domains - foo.com</i> • <i>IP addresses - 11.11.11.12</i>
✓	Sample Data	Selection of indicator data which can be used for functional testing. The enrichment must return data on these indicators. Indicators of all supported types must be provided.
✓	Enrichment Logo	Logo displayed to ThreatStream users on the enrichment tile. Images can be in JPG, PNG, or SVG format. Transparent backgrounds are strongly recommended. <i>Maximum file size: 300 KB</i> <i>Recommended image size: 250px by 80px</i>
✓	Enrichment Visibility	Visibility setting for the enrichment—whether it should be private to your organization or available to the Anomali community.
✓	Enrichment Outline	<i>For Anomali purposes only.</i> To expedite the enrichment review process, provide an overview of what the enrichment does for Anomali engineers.
✓	Vendor Website	URL of your homepage.
✓	Vendor Product Page	URL of the page dedicated to the product relevant to the enrichment.
✓	Support email address	Email address of your support team.
✓	Joint Anomali Product Page	If applicable, provide the URL of the joint marketing page for the enrichment.
Enrichment Authentication Information		

✓	Requirement	Description
✓	Authentication Required?	Please specify <i>yes</i> or <i>no</i> .
✓	Required Credentials	List the credentials that users must have in order to activate the enrichment. <i>Examples:</i> <ul style="list-style-type: none"> • <i>Certificate</i> • <i>UID or CID</i> • <i>API Key</i> • <i>Password</i>
✓	Valid Enrichment Credentials	Credentials which Anomali can use to activate and test the enrichment. These are needed for Anomali to test and certify your enrichment.
✓	Portal Credentials	Credentials and any other information relevant to accessing the portal from which your data originates. As part of certification, Anomali must check data returned by the enrichment against data from the source.
SDK Information		
✓	SDK Version	Version number of AnomaliEnrichments SDK you used to develop the enrichment.
Enrichment Collateral		
Note: These items must be compressed into a single ZIP file and attached to the email.		
✓	Enrichment Bundle	Final TGZ enrichment bundle. See "Creating Enrichment Bundles" on page 32 for more information.
✓	Enrichment Documentation	Documentation for your enrichment in PDF format. See "Providing Documentation for Your Enrichment" on the next page for more information.

Anomali performs code review and sanity checks on the enrichment. You will be notified if further changes are required. If no further changes are required, Anomali certifies your enrichment. After certifying the enrichment, Anomali installs the enrichment on ThreatStream production, making it available to your organization only or the Anomali community.

Providing Documentation for Your Enrichment

As part of the enrichment development process, you must provide documentation that instructs users on activating and using your enrichment. Documentation must include the following:

- Supported observables types
- Screenshots of the enrichment on the Explore pivoting tool and observable details pages
- A full list of transformations
- Activation instructions
- Links to more information on your organization and the enrichment
- (If updating your enrichment) A change-log describing updates to your enrichment in the latest release

See "[Sample Documentation](#) " on [page 52](#) for an example.

Chapter 8: Updating Enrichment Bundles

The Anomali Enrichments SDK enables in-place updates of enrichment bundles. You can use the guidelines in this section to make necessary changes to your enrichment and upload it, thus replacing the previous version. When enrichments are updated in adherence with these guidelines, all subscriptions remain active and do not require user intervention.

Updates can involve changes to source python scripts, enrichment icons, or any of the following metadata items:

- title
- transform_set_name
- description
- credentials

Note: See "[Updating Enrichment Credentials](#)" below before modifying your credentials.

- icon_display
- icon_thumbnail
- version
- transforms
- author
- long_description
- license
- source_url

Updating Enrichment Credentials

The Enrichments SDK supports adding new credential fields only when you update your enrichments. To ensure continuity of service for existing users or your

enrichment, credential fields added to enrichments must be optional.

To update your enrichment credentials, add the new optional credential field to your metadata file and set the `update_creds` parameter to `true`.

The following example illustrates the successful addition of a new credential field to an enrichment metadata file.

```
"version": "1.0.2",  
"update_creds": true,  
"credentials": [  
  {  
    "name": "api-key",  
    "description": "API Key for enrichment service",  
    "label": "API Key",  
    "required": true,  
    "sensitive": true  
  },  
  {  
    "name": "api-url",  
    "description": "API URL for enrichment service",  
    "label": "URL",  
    "required": false,  
    "sensitive": false  
  }  
],
```

The second credential field—`api-url`—is new. The `required` parameter for the credential is set to `false`. Additionally, the metadata file contains the `update_creds` parameter.

Requirements

In order to update enrichments in-place, you *must* adhere to these requirements:

- `app_name` values must be identical to the values of previously installed enrichments
- `version` must be updated to reflect the latest version

Example: 1.0.0 to 1.1.0 or 1.0.0 to 2.0.0

After making necessary changes to your bundle and ensuring that the above requirements are met, you can test your enrichment in the environment appropriate to your deployment. When testing is complete, ThreatStream Cloud users can submit updated enrichment to Anomali using the guidelines in ["Submitting ThreatStream Cloud Enrichments for Certification"](#) on page 43

Chapter 9: Troubleshooting Your Enrichment

If your enrichment does not work as expected, consult the information in this section to troubleshoot the issue before sending an email to Anomali support. This section also provides instructions on gathering relevant information that must be included in support emails.

To troubleshoot your enrichment:

1. **Ensure your script does not contain exceptions.**

Use the instructions in ["Testing Your Python Script" on page 29](#) to execute the script on your local machine. Inspect the output of the script and verify that the "exceptions" field is empty.

The following is an example of valid output that does not contain exceptions:

```
python abc_enrichment.py enrichIP 145.239.87.21 --credentials {"api_key\":"y3*****T8r\"}
{"widgets":[{"chartName":"Geo-map","widgetType":"Chart","highchartsJson":{"series\":[{"borderColor\":"#A0A0A0\","nullColor\":"rgba(200, 200, 200, 0.3)\","name\":"Basemap\","showInLegend\":"*****"itemType":"String","itemValue":"A honeypot probability score ranging from 0 (not a honeypot) to 1.0 (is a honeypot)"}, {"widgetType":"Text","lineBreakEnding":true}], "exceptions":[], "messages":[]}]
Process finished with exit code 0
```

2. **Ensure your enrichment is activated.**

Verify that the correct credentials were used to activate your enrichment.

3. **Ensure all transforms perform as expected.**

Verify that the issue is not caused by issues with the python script.

Sending Enrichment SDK Support Emails to Anomali

If your issue was not solved by following the troubleshooting steps above, send an email to enrichsdksupport@anomali.com that contains the information detailed in this section. The required information differs based on the ThreatStream form factor for which you developed the enrichment.

For ThreatStream Cloud enrichments, support emails must contain the following information:

- **ThreatStream User:** Email address associated with the account on the ThreatStream staging instance used for testing.
- **ThreatStream Organization:** Name of the organization registered on the ThreatStream staging instance.

Example: *anomali.com*

- **Enrichment Bundle Name:** Name of the enrichment displayed on the Integrations screen within settings.
- **Enrichment Version:** Version number specified in the metadata file.

Example: *1.0.1*

- **Enrichment Bundle:** Attach your enrichment bundle in .tar format to the email.
- **Credentials:** Credentials that Anomali can use to activate your enrichment. The credentials you include can be temporary. However, you must ensure that the credentials you include give Anomali full access to the services offered by the enrichment.
- **Input entity type and value:** Type of observable on which the enrichment runs and an example observable value for which it should exhibit expected behavior.

Example: *IP, 136.56.103.201*

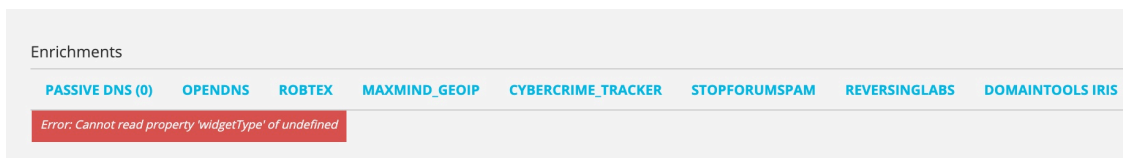
- **Backend Response:** Include the full text of the error message in addition to screenshots of the error as displayed on the user interface and the backend. Use the developer tools on your browser and locate the response for the following endpoint:

`https://svlpartner-optic-api.threatstream.com/api/v1/integration_package/transform/`

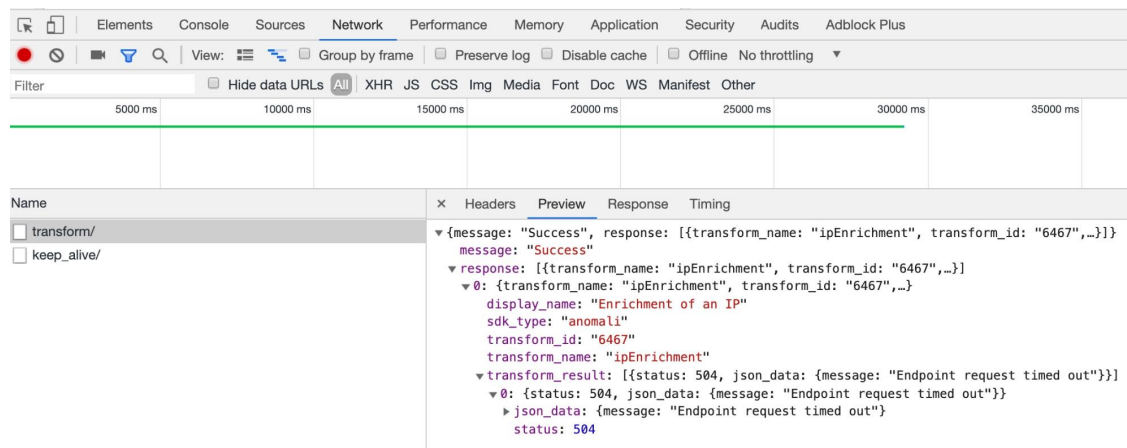
Example error text:

```
{"message": "Success", "response": [{"transform_name": "ipEnrichment",  
  "transform_id": "1111", "transform_result": [{"status": 504, "json_data":  
    {"message": "Endpoint request timed out"}}], "sdk_type": "anomali", "display_  
  name": "Enrichment of an IP"}]}
```

Example UI screenshot:



Example backend screenshot:



- **Expected Behavior:** Screenshot that depicts a rendering of the expected behavior on your portal.
- **Portal Credentials:** (Optional) Credentials for accessing the portal that provides enrichment data.
- **Client Details:** Browser, browser version, OS, and OS version on which the issue was experienced.

If you cannot provide the details requested to adequately describe the issue, email enrichsdksupport@anomali.com for further support.

Appendix A: Sample Documentation

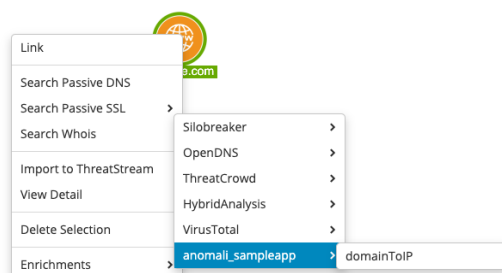
This appendix contains documentation for the Anomali Sample App, which is included in the Anomali Enrichments SDK bundle. You can use it as a template to guide documentation creation for your enrichment.

Enriching Data with the Anomali Sample App

What's New in Anomali Sample APP v2:

- Added a Domain to Email address transformation.
- General bug fixes.

When activated, the Anomali Sample App enrichment can provide data enrichments from Virus Total for domains, email, addresses, and IP addresses. You can leverage Anomali Sample App data on the Explore pivoting tool after activation.



The Anomali Sample App also displays enrichment data is available on observable details pages. For example, the following displays the hostnames to which an IP address resolves.



Enrichments

SAMPLEAPP ENRICHMENT

IP Enrichment for 104.27.158.93

Resolutions

25 1 - 25 of 924 items

Hostname	Last Resolved
06903609.cn	2019-01-16 20:20:04
12fandub.mt	2018-12-20 17:07:04
13870.com	2018-04-19 10:04:53
1742919.sop	2016-11-17 00:00:00
2000k6.tw	2018-11-20 23:54:06
2grktgnet.mt	2018-09-23 23:28:28
33115286.cn	2018-09-26 12:35:18
33d3d6.ga	2018-11-19 19:52:10

To read more about Anomali, see www.anomali.com

The Anomali Sample App enables the following data transformations:


- **domaintoIP**: returns IP addresses that resolve to the domain.
- **IPtodomain**: returns domains that resolve to the IP address.
- **domaintoEmail**: returns email addresses associated with the domain.

To activate the Anomali Sample App enrichment:

1. Log in to the ThreatStream user interface.
2. In the top navigation bar, click the settings icon and then **Integrations**.
3. Click **Set Up** in the sampleapp-anomali box.
4. Click **I have already registered** and enter your Virus Total API Key.

SAMPLEAPP ENRICHMENT

Transform set of Anomali enrichments for VirusTotal



Author	Version
	2.0.0
Source	License

For enrichment credentials:
Please contact vendor

I have Already Registered

Note: If you do not have a Virus Total API Key see the following URL for more information: <https://support.virustotal.com/hc/en-us/articles/115002088769-Please-give-me-an-API-key>

5. Click **Activate**.

If errors occur, contact support@anomali.com for assistance.

The Anomali Sample App enrichment is now active.