mstic Notebooklets Documentation

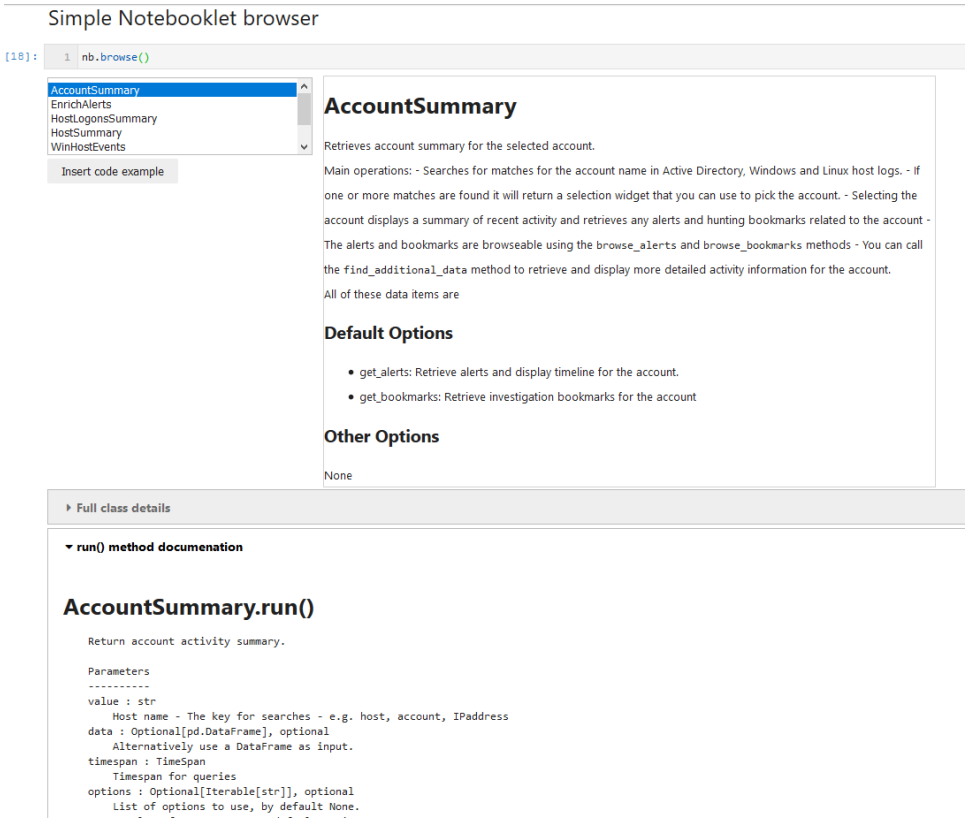
Release 0.2.0

Ian Hellen

Sep 02, 2022

Contents

msticnb is a companion package to msticpy. It is designed to be used in Jupyter notebooks by security operations engineers and analysts, to give them quick access to common notebook patterns such as retrieving summary information about a host or IP address.



Each notebooklet is equivalent to multiple cells and many lines of code in a traditional notebook. You can import and run a notebooklet with two lines of code (or even 1 line, if you are impatient). Typically, the input parameters to a notebooklet will be an identifier (e.g. a host name) and a time range (over which to query data). Some notebooklets (primarily packaged analytics) will take a pandas DataFrame as input.

```
host_summary = nb.nblts.azsent.host.HostSummary()
host_sum_rslt = host_summary.run(value="Msticalertswin1", timespan=time_span)
```

You can create your own notebooklets and use them in the same framework as the ones already in the package.

Read on to find out more about using and creating notebooklets.

CHAPTER 1

Introduction and Usage

1.1 Notebooklets Overview

1.1.1 What are notebooklets?

Notebooklets are collections of notebook cells that implement some useful reusable sequence. They are extensions of, and build upon the msticpy package and are design to streamline authoring of Jupyter notebooks for CyberSec hunters and investigators. The goal of notebooklets is to replace repetitive and lengthy boilerplate code in notebooks for common operations.

Some examples are:

- Get a host summary for a named host (IP address, cloud registration information, recent alerts)
- Get account activity for an account (host and cloud logons and failures, summary of recent activity)
- Triage alerts with Threat Intel data (prioritize your alerts by correlating with Threat intel sources)

1.1.2 Intended Audience

- Cyber security investigators and hunters using Jupyter notebooks for their work
- Security Ops Center (SOC) engineers/SecDevOps building reusable notebooks for SOC analysts

1.1.3 Why did we create notebooklets?

- Notebook code can quickly become complex and lengthy:
  - obscures the information you are trying to display
  - can be intimidating to non-developers
- Code in notebook code cells is not easily re-useable:
  - You can copy and paste but how do you sync changes back to the original notebook?

  - Difficult to discover code snippets in notebooks
- Notebook code is often fragile:
  - Often not parameterized or modular
  - Code blocks are frequently dependent on global values assigned earlier
  - Output data is not in any standard format
  - Difficult to test

1.1.4 Why aren't these part of msticpy?

- Msticpy aims to be platform-independent, whereas most if not all notebooklets assume a data schema that is specific to their data provider/SIEM.
- Msticpy is mostly for discrete functions such as data acquisition, analysis and visualization. Msticnb implements common SOC scenarios using this functionality.

1.1.5 Traditional Notebook vs. one using a Notebooklets

The notebook on the left is using mostly inline code (occupying more than 50% of the notebook). The one on the right is using a single notebooklet with only 3 or 4 lines of code.

1.1.6 Characteristics of Notebooklets

- They have one or small number of entry points/methods (typically a "run" method)
- They are parametrizable (e.g. you can supply hostname, IP Address, time range, etc.) and they may have runtime options to allow to skip unwanted processing or include optional processing
- They can query, process or visualize data (or any combination)
- They return a package of results that can be used later in the notebook
- The code can be imported into a notebook cell to be modified, if needed.

Limitations

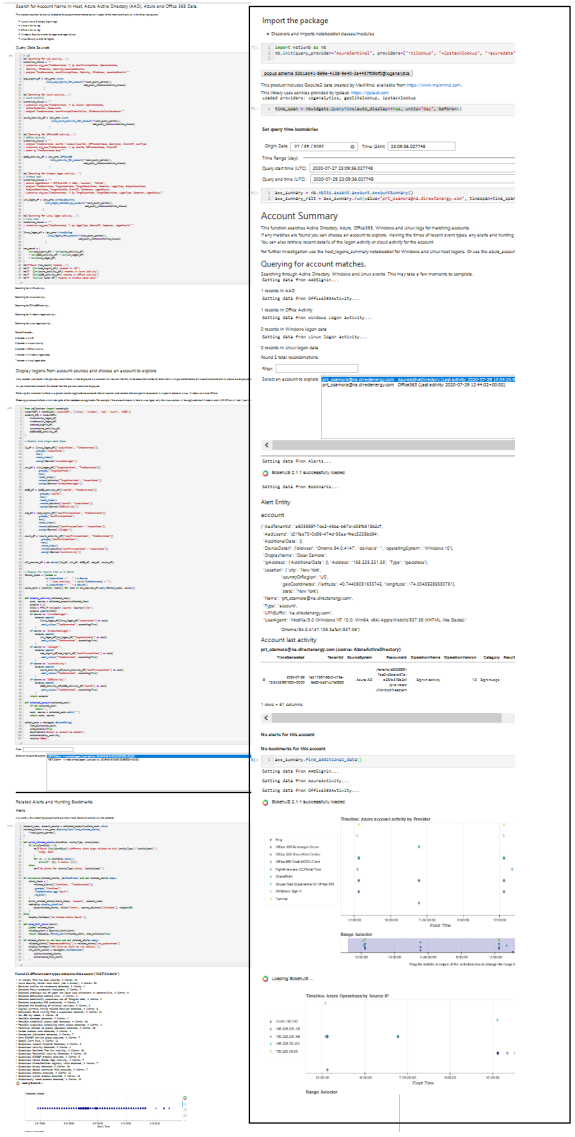
- They are normally specific to a data backend/SIEM since the data schema and layout varies between SIEM vendors.
- Notebooklet code layout is typically more complex than standard notebook code

1.2 Using Notebooklets

For a more detailed explanation of these steps and illustration of other features see the Notebooklets notebook

1.2.1 Install the Package

```
pip install msticnb
```



1.2.2 Import and initialize

```
[4]: 1 import msticnb as nb

7 notebooklets loaded.
```

The init method loads data drivers and data providers relevant to the the chosen data platform.

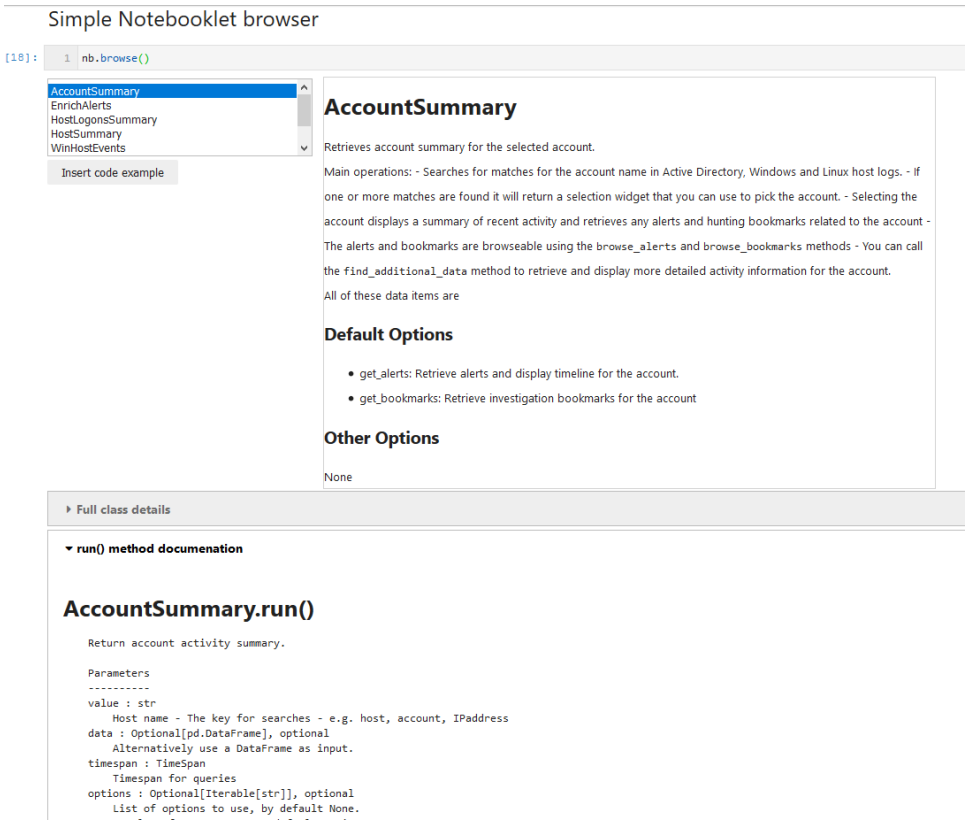
```
[20]: 1 nb.init(query_provider="AzureSentinel")

Using Open PageRank. See https://www.domcop.com/openpagerank/what-is-openpagerank
Loaded providers: LogAnalytics, geolitelookup, tilookup
```

1.2.3 Pick a notebooklet to use

You can pick a notebooklet from the commandline, using autocompletion. You can also search for a notebooklet using keywords and text from the notebooklet name and documentation.

The easiest way is using the nb.browse() method. This lists all of the available notebooklets and displays documentation, usage information and sample code snippet for each.



1.2.4 Instantiate the notebooklet and execute "run"

Notebooklets usually have a single `run` method, which is the entry point for the notebooklet. A notebooklet might have additional methods to do further drill-down, data retrieval, visualization or other operations once the run method has completed. Run typically requires parameters such as a host or account identifier and a time range over which to perform the operations.

```
[23]: 1 win_host_events = nb.nblts.azsent.host.WinHostEvents()
      2 timespan = TimeSpan(start="2020-05-07 00:10")
      3 win_host_events_rslt = win_host_events.run(value="MSTICAlertsWin1", timespan=timespan)
```

Host Security Events Summary

This shows a summary of security events for the host. These are grouped by EventID and Account.

Data and plots are stored in the result class returned by this function.

```
Getting data from SecurityEvent...
```

The notebooklet displays output directly to the notebook (although this can be suppressed) - showing text, data tables and visualizations. This data is all saved to a Results object. The data items are simple properties of this results object, for example, DataFrames, plots, or simple Python dictionaries. You can access these individually and you can just display the results object using IPython display() or just typing its name into and emtpy cell and running the cell.

1.2.5 View extended help for a notebooklet

You can access detailed documentation from any notebooklet class or instance using the show_help() method. This help includes a high-level description and usage information (parameters, available methods, options). It also describes the major output sections that will be displayed and the the contents of the return results.

Note: the contents of this help are also displayed in the notebooklet browser shown earlier.

1.3 Current Notebooklets

1.3.1 AccountSummary

Retrieves account summary for the selected account.

Main operations:

- Searches for matches for the account name in Active Directory, Windows and Linux host logs.
- If one or more matches are found it will return a selection widget that you can use to pick the account.
- Selecting the account displays a summary of recent activity and retrieves any alerts and hunting bookmarks related to the account
- The alerts and bookmarks are browsable using the browse_alerts and browse_bookmarks methods
- You can call the find_additional_data method to retrieve and display more detailed activity information for the account.

[23]:
```
win_host_events = nb.nblts.azsent.host.WinHostEvents()
timespan = TimeSpan(start="2020-05-07 00:10")
win_host_events_rslt = win_host_events.run(value="MSTICAlertsWin1", timespan=timespan)
```

Host Security Events Summary

This shows a summary of security events for the host. These are grouped by EventID and Account.

Data and plots are stored in the result class returned by this function.
Getting data from SecurityEvent...

Summary of Security Events on host

This is a summary of Security events for the host (excluding process creation and account logon - 4688, 4624, 4625).

Yellow highlights indicate account with highest event count for an EventID.

Activity	DWM-1	DWM-2	IUSR	LOCAL SERVICE	MSTICAdmin	MSTICAlertsWin1$	NETWORK SERVICE	No Account	SYSTEM	lan
1100 - The event logging service has shut down.								1.000000		
4608 - Windows is starting up.								1.000000		
4616 - The system time was changed.				2.000000						
4625 - An account failed to log on.										2.000000
4634 - An account was logged off.		4.000000			12.000000					2.000000
4647 - User initiated logoff.					2.000000					
4648 - A logon was attempted using explicit credentials.						10.000000				
4672 - Special privileges assigned to new logon.	2.000000	2.000000	1.000000	1.000000	14.000000		1.000000		60.000000	
4720 - A user account was created.					2.000000					
4722 - A user account was enabled.					2.000000					
4724 - An attempt was made to reset an account's password.					4.000000					
4726 - A user account was deleted.					2.000000					
4728 - A member was added to a security-enabled global group.					2.000000					
4729 - A member was removed from a security-enabled global group.					2.000000					
4732 - A member was added to a security-enabled local group.					4.000000					
4733 - A member was removed from a security-enabled local group.					3.000000					
4738 - A user account was changed.					5.000000					
4776 - The domain controller attempted to validate the credentials for an account.								18.000000		
4798 - A user's local group membership was enumerated.					14.000000	652.000000				
4799 - A security-enabled local group membership was enumerated					4.000000	236.000000				
4826 - Boot Configuration Data loaded.								1.000000		
4902 - The Per-user audit policy table was created.								1.000000		
4904 - An attempt was made to register a security event source.						1.000000				
4907 - Auditing settings on object were changed.						198.000000				
5024 - The Windows Firewall Service has started successfully.								1.000000		
5033 - The Windows Firewall Driver has started successfully.								1.000000		
5058 - Key file operation.								159.000000		
5059 - Key migration operation.				1.000000		206.000000				
5061 - Cryptographic operation.								159.000000		
8001								1.000000		
8002 - A process was allowed to run.					192.000000				12.000000	

Summary of Account Management Events on host

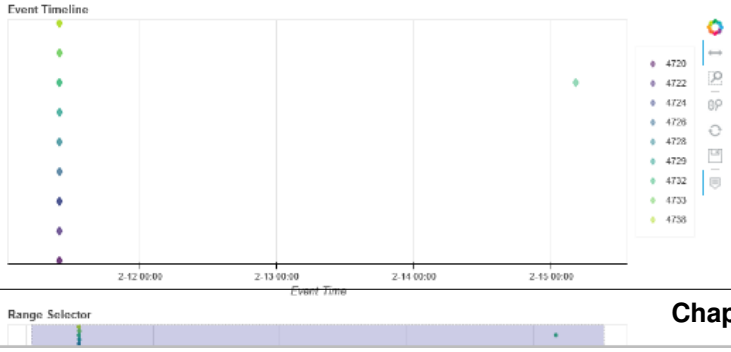
This shows the subset of events related to account management, for example, creation/deletion of accounts, changes to group membership, etc.

Yellow highlights indicate account with highest event count.

Activity	MSTICAdmin
4720 - A user account was created.	2
4722 - A user account was enabled.	2
4724 - An attempt was made to reset an account's password.	4
4726 - A user account was deleted.	2
4728 - A member was added to a security-enabled global group.	2
4729 - A member was removed from a security-enabled global group.	2
4732 - A member was added to a security-enabled local group.	4
4733 - A member was removed from a security-enabled local group.	3
4738 - A user account was changed.	5

Timeline of Account Management Events on host

BokehJS 2.1.1 successfully loaded.



```
[14]: 1 nb.nblts.azsent.host.HostSummary.show_help()
```

Notebooklet Class - HostSummary

HostSummary Notebooklet class.

Queries and displays information about a host including:

- IP address assignment
- Related alerts
- Related hunting/investigation bookmarks
- Azure subscription/resource data.

Default Options

- heartbeat: Query Heartbeat table for host information.
- azure_net: Query AzureNetworkAnalytics table for host network topology information.
- alerts: Query any alerts for the host.
- bookmarks: Query any bookmarks for the host.
- azure_api: Query Azure API for VM information.

Other Options

None

Display Sections

Host Entity Summary

This shows a summary data for a host. It shows host properties obtained from OMS Heartbeat and Azure API. It also lists Azure
and plots are stored in the result class returned by this function.

Timeline of related alerts

Each marker on the timeline indicates one or more alerts related to the host.

1.3.2 EnrichAlerts

Alert Enrichment Notebooklet Class.

Enriches Azure Sentinel alerts with Threat Intelligence data.

1.3.3 HostLogonsSummary

Host Logons Summary Notebooket class.

Queries and displays information about logons to a host including:

- Summary of successful logons
- Visualizations of logon event times
- Geolocation of remote logon sources
- Visualizations of various logon elements depending on host type
- Data on users with failed and successful logons

1.3.4 HostSummary

HostSummary Notebooklet class.

Queries and displays information about a host including:

- IP address assignment
- Related alerts
- Related hunting/investigation bookmarks
- Azure subscription/resource data.

1.3.5 WinHostEvents

Windows host Security Events Notebooklet class.

Queries and displays Windows Security Events including:

- All security events summary
- Extracting and displaying account management events
- Account management event timeline
- Optionally parsing packed event data into DataFrame columns

Process (4688) and Account Logon (4624, 4625) are not included in the event types processed by this module.

1.3.6 IpAddressSummary

Retrieves common data about an IP Address including:

- Tries to determine IP address is external or internal (i.e. owned by the organization)
- Azure Heartbeat, Network Analytics or VMComputer records
- Geo-IP and Whois data

- Threat intel reports
- Related alerts and hunting bookmarks
- Network flows involving IP address
- Azure activity (e.g. sign-ins) originating from IP address

1.3.7 NetworkFlowSummary

Network Flow Summary Notebooklet class.

Queries network data and plots time lines for network traffic to/from a host or IP address.

- Plot flows events by protocol and direction
- Plot flow count by protocol
- Display flow summary table
- Display flow summary by ASN
- Display results on map

1.3.8 TemplateNB

Template Notebooklet class.

A code template for creating additional notebooklets.

1.3.9 See Also

msticpy documentation

CHAPTER 2

Notebooklet details

2.1 Notebooklets Details

2.1.1 Notebooklet Class - AccountSummary

Retrieves account summary for the selected account.

Main operations:

- Searches for matches for the account name in Active Directory,

Windows and Linux host logs.

- If one or more matches are found it will return a selection

widget that you can use to pick the account.

- Selecting the account displays a summary of recent activity and

retrieves any alerts and hunting bookmarks related to the account

- The alerts and bookmarks are browseable using the `browse_alerts`

and `browse_bookmarks` methods

- You can call the `get_additional_data` method to retrieve and

display more detailed activity information for the account.

All of the returned data items are stored in the results class

as entities, pandas DataFrames or Bokeh visualizations.

Run help(nblt) on the notebooklet class to see usage.

Run help(result) on the result class to see documentation of its

properties.

Run the print_options() method on either the notebooklet or

results class to see information about the `options` parameter

for the run() method.

Default Options

- get_alerts: Retrieve alerts and display timeline for the account.
- get_bookmarks: Retrieve investigation bookmarks for the account

Other Options

None

Display Sections

Account Summary

This function searches Active Directory, Azure, Office365, Windows and Linux logs for matching accounts. If any matches are found you can choose an account to explore, viewing the times of recent event types, any alerts and hunting bookmarks that relate to the account name. You can also retrieve recent details of the logon activity or cloud activity for the account. For further investigation use the host_logons_summary notebooklet for Windows and Linux host logons.

Host logon attempt timeline

Hover over each timeline event to see details.

IP Address details summary

Number of operations detected by IP Address. The table shows WhoIs ASN Description and Country Code. If UserAgent is contained in the data, operations are also grouped by this.

Querying for account matches.

Searching through Active Directory, Windows and Linux events. This may take a few moments to complete.

Summary of azure activity for AAD, Azure resource and O365

Shows the total number of operations, the list of unique operations, the list of unique resource IDs and the first and last operation recorded in the selected time range. The data is grouped by: - Data source - User - Type - Azure activity type/source - Client IP Address - Application resource provider - User type

Summary of host logon activity.

Shows the total number of logons attempts by host. FailedLogons shows the breakdown of successfully and failed logons. IPAddresses is a list of distinct source IP addresses for the logons. LogonTypeCount breaks down the logon type used by count. First and LastLogon shows the earliest and latest logons on each host by this account in the selected time range.

Results Class

AccountSummaryResult

Account Summary Result.

Attributes

- account_activity : pd.DataFrame
  DataFrame of most recent activity.
- account_selector : msticpy.nbtools.nbwidgets.SelectString
  Selection widget for accounts.
- related_alerts : pd.DataFrame
  Alerts related to the account.
- alert_timeline : LayoutDOM
  Timeline of alerts.
- related_bookmarks : pd.DataFrame
  Investigation bookmarks related to the account.
- host_logons : pd.DataFrame
  Host logon attemtps for selected account.
- host_logon_summary : pd.DataFrame
  Host logon summary for selected account.
- azure_activity : pd.DataFrame
  Azure Account activity for selected account.
- account_activity_summary : pd.DataFrame
  Azure activity summary.
- azure_timeline_by_provider : LayoutDOM
  Azure activity timeline grouped by provider
- account_timeline_by_ip : LayoutDOM
  Host or Azure activity timeline by IP Address.
- azure_timeline_by_operation : LayoutDOM
  Azure activity timeline grouped by operation
- ip_address_summary : pd.DataFrame
  Summary of IP address properties and usage for the current activity.
- ip_all_data : pd.DataFrame
  Full details of operations with IP WhoIs and GeoIP data.

Methods

Instance Methods

__init__(self, *args,* *kwargs)

Initialize the Account Summary notebooklet.

az_activity_timeline_by_ip(self)
Display Azure activity timeline by IP address.

az_activity_timeline_by_operation(self)
Display Azure activity timeline by operation.

az_activity_timeline_by_provider(self)
Display Azure activity timeline by provider.

browse_accounts(self) -> msticpy.nbtools.nbwidgets.select_item.SelectItem
Return the accounts browser/viewer.

browse_alerts(self) -> msticpy.nbtools.nbwidgets.select_alert.SelectAlert
Return alert browser/viewer.

browse_bookmarks(self) -> msticpy.nbtools.nbwidgets.select_item.SelectItem
Return bookmark browser/viewer.

display_alert_timeline(self)
Display the alert timeline.

get_additional_data(self) -> pandas.core.frame.DataFrame
Find additional data for the selected account.

get_geoip_map(self)
Return Folium map of IP activity.

host_logon_timeline(self)
Display IP address summary.

run(self, value: Any = None, data: Optional[pandas.core.frame.DataFrame] = None, timespan: Optional[msticpy.common.timespan.TimeSpan] = None, options: Optional[Iterable[str]] = None, **kwargs) -> msticnb.nb.azsent.account.account_summary.AccountSummaryResult
Return account activity summary.

show_ip_summary(self)
Display Azure activity timeline by operation.

Inherited methods

check_table_exists(self, table: str) -> bool
Check to see if the table exists in the provider.

check_valid_result_data(self, attrib: str = None, silent: bool = False) -> bool
Check that the result is valid and `attrib` contains data.

get_methods(self) -> Dict[str, Callable[[Any], Any]]
Return methods available for this class.

get_pivot_run(self, get_timespan: Callable[[], msticpy.common.timespan.TimeSpan])
Return Pivot-wrappable run function.

get_provider(self, provider_name: str)
Return data provider for the specified name.

list_methods(self) -> List[str]
Return list of methods with descriptions.

run_nb_func(self, nb_func: Union[str, msticnb.notebooklet_func.NBFunc], **kwargs)
Run the notebooklet function and return the results.

run_nb_funcs(self)
Run all notebooklet functions defined for the notebooklet.

Other Methods

add_nb_function(nb_func: Union[str, msticnb.notebooklet_func.NBFunc], **kwargs)
Add a notebooklet function to the class.

all_options() -> List[str]
Return supported options for Notebooklet run function.

default_options() -> List[str]
Return default options for Notebooklet run function.

description() -> str
Return description of the Notebooklet.

entity_types() -> List[str]
Entity types supported by the notebooklet.

get_help(fmt='html') -> str
Return HTML document for class.

get_settings(print_settings=True) -> Optional[str]
Print or return metadata for class.

import_cell()
Import the text of this module into a new cell.

keywords() -> List[str]
Return search keywords for Notebooklet.

list_options() -> str
Return options document for Notebooklet run function.

match_terms(search_terms: str) -> Tuple[bool, int]
Search class definition for `search_terms`.

name() -> str
Return name of the Notebooklet.

print_options()
Print options for Notebooklet run function.

result [property] Return result of the most recent notebooklet run.

show_help()
Display Documentation for class.

silent [property] Get the current instance setting for silent running.

<hr>

`run` function documentation

Return account activity summary.

Parameters

value [str] Account name to search for.

data [Optional[pd.DataFrame], optional] Not used.

timespan [TimeSpan] Timespan for queries

options [Optional[Iterable[str]], optional] List of options to use, by default None. A value of None means use default options. Options prefixed with "+" will be added to the default options. To see the list of available options type *help(cls)* where "cls" is the notebooklet class or an instance of this class.

account_types [Iterable[AccountType], Optional] A list of account types to search for, by default all types.

Returns

AccountSummaryResult Result object with attributes for each result type.

Raises

MsticnbMissingParameterError If required parameters are missing

Default Options

- get_alerts: Retrieve alerts and display timeline for the account.
- get_bookmarks: Retrieve investigation bookmarks for the account

Other Options

None

2.1.2 Notebooklet Class - EnrichAlerts

Alert Enrichment Notebooklet Class.

Enriches Azure Sentinel alerts with TI data.

Display Sections

Results Class

TIEnrichResult

Template Results.

Attributes

- enriched_results : pd.DataFrame
  Alerts with additional TI enrichment
- picker : SelectAlert
  Alert picker

Methods

Instance Methods

__init__

__init__(self, data_providers: Optional[<msticnb.data_providers.SingletonDecorator object at 0x0000023FAFA3A6A0>] = None, **kwargs)
Initialize a new instance of the notebooklet class.

run

run(self, value: Optional[str] = None, data: Optional[pandas.core.frame.DataFrame] = None, timespan: Optional[msticpy.common.timespan.TimeSpan] = None, options: Optional[Iterable[str]] = None, **kwargs) -> msticnb.nb.azsent.alert.ti_enrich.TIEnrichResult
Return an enriched set of Alerts.

Inherited methods

check_table_exists

check_table_exists(self, table: str) -> bool
Check to see if the table exists in the provider.

check_valid_result_data

check_valid_result_data(self, attrib: str = None, silent: bool = False) -> bool
Check that the result is valid and `attrib` contains data.

get_methods

get_methods(self) -> Dict[str, Callable[[Any], Any]]
Return methods available for this class.

get_pivot_run

get_pivot_run(self, get_timespan: Callable[[], msticpy.common.timespan.TimeSpan])
Return Pivot-wrappable run function.

get_provider

get_provider(self, provider_name: str)
Return data provider for the specified name.

list_methods

list_methods(self) -> List[str]
Return list of methods with descriptions.

run_nb_func

run_nb_func(self, nb_func: Union[str, msticnb.notebooklet_func.NBFunc], **kwargs)
Run the notebooklet function and return the results.

run_nb_funcs

run_nb_funcs(self)
Run all notebooklet functions defined for the notebooklet.

Other Methods

add_nb_function

add_nb_function(nb_func: Union[str, msticnb.notebooklet_func.NBFunc], **kwargs)
Add a notebooklet function to the class.

all_options

all_options() -> List[str]
Return supported options for Notebooklet run function.

default_options

default_options() -> List[str]
Return default options for Notebooklet run function.

description

description() -> str
Return description of the Notebooklet.

entity_types

entity_types() -> List[str]
Entity types supported by the notebooklet.

get_help

get_help(fmt='html') -> str
Return HTML document for class.

get_settings

get_settings(print_settings=True) -> Optional[str]
Print or return metadata for class.

import_cell

import_cell()
Import the text of this module into a new cell.

keywords

keywords() -> List[str]
Return search keywords for Notebooklet.

list_options

list_options() -> str
Return options document for Notebooklet run function.

match_terms

match_terms(search_terms: str) -> Tuple[bool, int]
Search class definition for `search_terms`.

name

name() -> str
Return name of the Notebooklet.

print_options

print_options()
Print options for Notebooklet run function.

result

result [property] Return result of the most recent notebooklet run.

show_help

show_help()
Display Documentation for class.

silent

silent [property] Get the current instance setting for silent running.

<hr>

`run` function documentation

Return an enriched set of Alerts.

Parameters

timespan [TimeSpan] Timespan for queries

options [Optional[Iterable[str]], optional] List of options to use, by default None. A value of None means use default options. Options prefixed with "+" will be added to the default options. To see the list of available options type *help(cls)* where "cls" is the notebooklet class or an instance of this class.

value: Optional[str], optional If you want to filter Alerts based on a specific entity specify it as a string.

data: Optional[pd.DataFrame], optional If you have alerts in a DataFrame you can pass them rather than having the notebooklet query alerts.

Returns

TIEnrichResult Result object with attributes for each result type.

Raises

MsticnbMissingParameterError If required parameters are missing

MsticnbDataProviderError If data is not avaliable

Default Options

- TI: Uses TI to enrich alert data. Will use your primary TI providers.
- details: Displays a widget allowing you to see more detail about an alert.

Other Options

- secondary: Uses secondary TI providers in lookups.

2.1.3 Notebooklet Class - HostLogonsSummary

Host Logons Summary Notebooket class.

Queries and displays information about logons to a host including:

- Summary of sucessfull logons
- Visualizations of logon event times
- Geolocation of remote logon sources
- Visualizations of various logon elements depending on host type
- Data on users with failed and sucessful logons

Display Sections

Results Class

HostLogonsSummaryResult

Host Logons Summary Results.

Attributes

- logon_sessions: pd.DataFrame
  A Dataframe summarizing all sucessfull and failed logon attempts observed during the specified time period.
-
- logon_map: FoliumMap
  A map showing remote logon attempt source locations. Red points represent failed logons, green successful.

-
- plots: Dict
  A collection of Bokeh plot figures showing various aspects of observed logons. Keys are a descriptive name of the plot and values are the plot figures.

Methods

Instance Methods

__init__

__init__(self, data_providers: Optional[<msticnb.data_providers.SingletonDecorator object at 0x0000023FAFA3A6A0>] = None, **kwargs)
Initialize a new instance of the notebooklet class.

run

run(self, value: Any = None, data: Optional[pandas.core.frame.DataFrame] = None, timespan: Optional[msticpy.common.timespan.TimeSpan] = None, options: Optional[Iterable[str]] = None, **kwargs) -> msticnb.nb.azsent.host.host_logons_summary.HostLogonsSummaryResult
Return host summary data.

Inherited methods

check_table_exists

check_table_exists(self, table: str) -> bool
Check to see if the table exists in the provider.

check_valid_result_data

check_valid_result_data(self, attrib: str = None, silent: bool = False) -> bool
Check that the result is valid and `attrib` contains data.

get_methods

get_methods(self) -> Dict[str, Callable[[Any], Any]]
Return methods available for this class.

get_pivot_run

get_pivot_run(self, get_timespan: Callable[[], msticpy.common.timespan.TimeSpan])
Return Pivot-wrappable run function.

get_provider

get_provider(self, provider_name: str)
Return data provider for the specified name.

list_methods

list_methods(self) -> List[str]
Return list of methods with descriptions.

run_nb_func

run_nb_func(self, nb_func: Union[str, msticnb.notebooklet_func.NBFunc], **kwargs)
Run the notebooklet function and return the results.

run_nb_funcs

run_nb_funcs(self)
Run all notebooklet functions defined for the notebooklet.

Other Methods

add_nb_function

add_nb_function(nb_func: Union[str, msticnb.notebooklet_func.NBFunc], **kwargs)
Add a notebooklet function to the class.

all_options

all_options() -> List[str]
Return supported options for Notebooklet run function.

default_options

default_options() -> List[str]
Return default options for Notebooklet run function.

description

description() -> str
Return description of the Notebooklet.

entity_types

entity_types() -> List[str]
Entity types supported by the notebooklet.

get_help

get_help(fmt='html') -> str
Return HTML document for class.

get_settings

get_settings(print_settings=True) -> Optional[str]
Print or return metadata for class.

import_cell

import_cell()
Import the text of this module into a new cell.

keywords

keywords() -> List[str]
Return search keywords for Notebooklet.

list_options

list_options() -> str
Return options document for Notebooklet run function.

match_terms

match_terms(search_terms: str) -> Tuple[bool, int]
Search class definition for `search_terms`.

name

name() -> str
Return name of the Notebooklet.

print_options

print_options()
Print options for Notebooklet run function.

result

result [property] Return result of the most recent notebooklet run.

show_help

show_help()
Display Documentation for class.

silent

silent [property] Get the current instance setting for silent running.

<hr>

`run` function documentation

Return host summary data.

Parameters

value [str] Host name

data [Optional[pd.DataFrame], optional] Optionally pass raw data to use for analysis, by default None

timespan [TimeSpan] Timespan over which operations such as queries will be performed, by default None. This can be a TimeStamp object or another object that has valid *start*, *end*, or *period* attributes. Alternatively you can pass *start* and *end* datetime objects.

options [Optional[Iterable[str]], optional] List of options to use, by default None A value of None means use default options.

Returns

HostLogonsSummaryResults Result object with attributes for each result type.

Raises

MsticnbMissingParameterError If required parameters are missing

MsticnbDataProviderError If data is not avaliable

Default Options

- map: Display a map of logon attempt locations.
- timeline: Display a timeline of logon atttempts.
- charts: Display a range of charts depicting different elements of logon events.
- failed_success: Displays a DataFrame of all users with both successful and failed logons.

Other Options

None

2.1.4 Notebooklet Class - HostNetworkSummary

Host Network Summary Notebooket class.

Queries and displays information about network connections by a host including:

- Summary of network connections
- Visualizations of network events
- Geolocation of remote IP addresses
- Threat Intelligence enrichment of remote IP addresses

Display Sections

Results Class

HostNetworkSummaryResult

Host Network Summary Results.

Attributes

- flows: pd.DataFrame
  A Dataframe summarizing all network flows to and from a host.
- flow_matrix: LayoutDOM
  A plot of network traffic volumes from the host.
- flow_whois: pd.DataFrame
  Network flow data to and from the host enriched with WhoIs information about the IP address.
- flow_map: FoliumMap
  A map showing the location of all remote IP addresses communicating with the host.
- flow_ti: pd.DataFrame

Network flow data to and from the host enriched with Threat Intelligence results for the IP address.

Methods

Instance Methods

__init__

__init__(self, data_providers: Optional[<msticnb.data_providers.SingletonDecorator object at 0x0000023FAFA3A6A0>] = None, **kwargs)
Initialize a new instance of the notebooklet class.

run

run(self, value: Any = None, data: Optional[pandas.core.frame.DataFrame] = None, timespan: Optional[msticpy.common.timespan.TimeSpan] = None, options: Optional[Iterable[str]] = None, **kwargs) -> msticnb.nb.azsent.host.host_logons_summary.HostLogonsSummaryResult
Return host network data.

Inherited methods

check_table_exists

check_table_exists(self, table: str) -> bool
Check to see if the table exists in the provider.

check_valid_result_data

check_valid_result_data(self, attrib: str = None, silent: bool = False) -> bool
Check that the result is valid and `attrib` contains data.

get_methods

get_methods(self) -> Dict[str, Callable[[Any], Any]]
Return methods available for this class.

get_pivot_run

get_pivot_run(self, get_timespan: Callable[[], msticpy.common.timespan.TimeSpan])
Return Pivot-wrappable run function.

get_provider

get_provider(self, provider_name: str)
Return data provider for the specified name.

list_methods

list_methods(self) -> List[str]
Return list of methods with descriptions.

run_nb_func

run_nb_func(self, nb_func: Union[str, msticnb.notebooklet_func.NBFunc], **kwargs)
Run the notebooklet function and return the results.

run_nb_funcs

run_nb_funcs(self)
Run all notebooklet functions defined for the notebooklet.

Other Methods

add_nb_function

add_nb_function(nb_func: Union[str, msticnb.notebooklet_func.NBFunc], **kwargs)
Add a notebooklet function to the class.

all_options

all_options() -> List[str]
Return supported options for Notebooklet run function.

default_options

default_options() -> List[str]
Return default options for Notebooklet run function.

description

description() -> str
Return description of the Notebooklet.

entity_types

entity_types() -> List[str]
Entity types supported by the notebooklet.

get_help

get_help(fmt='html') -> str
Return HTML document for class.

get_settings

get_settings(print_settings=True) -> Optional[str]
Print or return metadata for class.

import_cell

import_cell()
Import the text of this module into a new cell.

keywords

keywords() -> List[str]
Return search keywords for Notebooklet.

list_options

list_options() -> str
Return options document for Notebooklet run function.

match_terms

match_terms(search_terms: str) -> Tuple[bool, int]
Search class definition for `search_terms`.

name

name() -> str
Return name of the Notebooklet.

print_options

print_options()
Print options for Notebooklet run function.

result

result [property] Return result of the most recent notebooklet run.

show_help

show_help()
Display Documentation for class.

silent

silent [property] Get the current instance setting for silent running.

<hr>

`run` function documentation

Return host network data.

Parameters

value [str] Host name

data [Optional[pd.DataFrame], optional] Optionally pass raw data to use for analysis, by default None

timespan [TimeSpan] Timespan over which operations such as queries will be performed, by default None. This can be a TimeStamp object or another object that has valid *start*, *end*, or *period* attributes. Alternatively you can pass *start* and *end* datetime objects.

options [Optional[Iterable[str]], optional] List of options to use, by default None A value of None means use default options.

Returns

HostNetworkSummaryResults Result object with attributes for each result type.

Raises

MsticnbMissingParameterError If required parameters are missing

MsticnbDataProviderError If data is not avaliable

Default Options

- map: Display a map of remote IP addresses communicating with the host.
- ti: Enrich network flow data with Threat Inteligence.
- whois: Enrich network flow data with WhoIs information.

Other Options

None

2.1.5 Notebooklet Class - HostSummary

HostSummary Notebooklet class.

Queries and displays information about a host including:

- IP address assignment
- Related alerts
- Related hunting/investigation bookmarks
- Azure subscription/resource data.

Default Options

- heartbeat: Query Heartbeat table for host information.
- azure_net: Query AzureNetworkAnalytics table for host network topology information.
- alerts: Query any alerts for the host.
- bookmarks: Query any bookmarks for the host.
- azure_api: Query Azure API for VM information.

Other Options

None

Display Sections

Host Entity Summary

This shows a summary data for a host. It shows host properties obtained from OMS Heartbeat and Azure API. It also lists Azure Sentinel alerts and bookmakrs related to to the host. Data and plots are stored in the result class returned by this function.

Timeline of related alerts

Each marker on the timeline indicates one or more alerts related to the host.

Host Entity details

These are the host entity details gathered from Heartbeat and, if applicable, AzureNetworkAnalytics and Azure management API. The data shows OS information, IP Addresses assigned the host and any Azure VM information available.

Results Class

HostSummaryResult

Host Details Results.

Attributes

- host_entity : msticpy.datamodel.entities.Host
  The host entity object contains data about the host such as name, environment, operating system version, IP addresses and Azure VM details. Depending on the type of host, not all of this data may be populated.
- related_alerts : pd.DataFrame
  Pandas DataFrame of any alerts recorded for the host within the query time span.
- alert_timeline:
  Bokeh time plot of alerts recorded for host.
- related_bookmarks: pd.DataFrame
  Pandas DataFrame of any investigation bookmarks relating to the host.

Methods

Instance Methods

__init__

__init__(self, data_providers: Optional[<msticnb.data_providers.SingletonDecorator object at 0x0000023FAFA3A6A0>] = None, **kwargs)
Initialize a new instance of the notebooklet class.

browse_alerts

browse_alerts(self) -> msticpy.nbtools.nbwidgets.select_alert.SelectAlert
Return alert browser/viewer.

run

run(self, value: Any = None, data: Optional[pandas.core.frame.DataFrame] = None, timespan: Optional[msticpy.common.timespan.TimeSpan] = None, options: Optional[Iterable[str]] = None, **kwargs) -> msticnb.nb.azsent.host.host_summary.HostSummaryResult
Return host summary data.

Inherited methods

check_table_exists

check_table_exists(self, table: str) -> bool
Check to see if the table exists in the provider.

check_valid_result_data

check_valid_result_data(self, attrib: str = None, silent: bool = False) -> bool
Check that the result is valid and `attrib` contains data.

get_methods

get_methods(self) -> Dict[str, Callable[[Any], Any]]
Return methods available for this class.

get_pivot_run

get_pivot_run(self, get_timespan: Callable[[], msticpy.common.timespan.TimeSpan])
Return Pivot-wrappable run function.

get_provider

get_provider(self, provider_name: str)
Return data provider for the specified name.

list_methods

list_methods(self) -> List[str]
Return list of methods with descriptions.

run_nb_func

run_nb_func(self, nb_func: Union[str, msticnb.notebooklet_func.NBFunc], **kwargs)
Run the notebooklet function and return the results.

run_nb_funcs

run_nb_funcs(self)
Run all notebooklet functions defined for the notebooklet.

Other Methods

add_nb_function

add_nb_function(nb_func: Union[str, msticnb.notebooklet_func.NBFunc], **kwargs)
Add a notebooklet function to the class.

all_options

all_options() -> List[str]
Return supported options for Notebooklet run function.

default_options

default_options() -> List[str]
Return default options for Notebooklet run function.

description

description() -> str
Return description of the Notebooklet.

entity_types

entity_types() -> List[str]
Entity types supported by the notebooklet.

get_help

get_help(fmt='html') -> str
Return HTML document for class.

get_settings

get_settings(print_settings=True) -> Optional[str]
Print or return metadata for class.

import_cell

import_cell()
Import the text of this module into a new cell.

keywords

keywords() -> List[str]
Return search keywords for Notebooklet.

list_options

list_options() -> str
Return options document for Notebooklet run function.

match_terms

match_terms(search_terms: str) -> Tuple[bool, int]
Search class definition for `search_terms`.

name

name() -> str
Return name of the Notebooklet.

print_options

print_options()
Print options for Notebooklet run function.

result

result [property] Return result of the most recent notebooklet run.

show_help

show_help()
Display Documentation for class.

silent

silent [property] Get the current instance setting for silent running.

<hr>

`run` function documentation

Return host summary data.

Parameters

value [str] Host name

data [Optional[pd.DataFrame], optional] Not used, by default None

timespan [TimeSpan] Timespan over which operations such as queries will be performed, by default None. This can be a TimeStamp object or another object that has valid *start*, *end*, or *period* attributes.

options [Optional[Iterable[str]], optional] List of options to use, by default None A value of None means use default options. Options prefixed with "+" will be added to the default options. To see the list of available options type *help(cls)* where "cls" is the notebooklet class or an instance of this class.

Other Parameters

start [Union[datetime, datelike-string]] Alternative to specifying timespan parameter.

end [Union[datetime, datelike-string]] Alternative to specifying timespan parameter.

Returns

HostSummaryResult Result object with attributes for each result type.

Raises

MsticnbMissingParameterError If required parameters are missing

Default Options

- heartbeat: Query Heartbeat table for host information.
- azure_net: Query AzureNetworkAnalytics table for host network topology information.
- alerts: Query any alerts for the host.
- bookmarks: Query any bookmarks for the host.
- azure_api: Query Azure API for VM information.

Other Options

None

2.1.6 Notebooklet Class - LogonSessionsRarity

Calculates the relative rarity of logon sessions.

It clusters the data based on process, command line and account.

Then calculates the rarity of the process pattern.

Then returns a result containing a summary of the logon sessions by rarity.

To see the methods available for the class and result class, run

cls.list_methods()

Default Options

None

Other Options

None

Display Sections

Calculate process rarity statistics for logon sessions

This first transforms the input data into features suitable for a clustering algorithm. It then clusters the data based on process, command line and account and calculates the rarity of the process pattern. It returns a result containing a summary of the logon sessions along with full results of the clustering. Methods available to view this data graphically include - list_sessions_by_rarity - table of sessions ordered by degree of rarity - plot_sessions_by_rarity - timeline plot of processes grouped by account and showing relative rarity of each process. - process_tree - a process tree of all processes or processes belonging to a single account.

Results Class

LogonSessionsRarityResult

Logon Sessions rarity.

Attributes

- process_clusters : pd.DataFrame
  Process clusters based on account, process, commandline and showing the an example process from each cluster
- processes_with_cluster : pd.DataFrame
  Merged data with rarity value assigned to each process event.
- session_rarity: pd.DataFrame
  List of sessions with averaged process rarity.

Methods

Instance Methods

__init__

__init__(self, **kwargs)
Initialize instance of LogonSessionRarity.

browse_events

browse_events(self)
Browse the events by logon session.

list_sessions_by_rarity

list_sessions_by_rarity(self)
Display sessions by process rarity.

plot_sessions_by_rarity

plot_sessions_by_rarity(self)
Display timeline plot of processes by rarity.

process_tree

process_tree(self, account: Optional[str] = None, session: Optional[str] = None)
View a process tree of current session.

run

run(self, value: Any = None, data: Optional[pandas.core.frame.DataFrame] = None, timespan: Optional[msticpy.common.timespan.TimeSpan] = None, options: Optional[Iterable[str]] = None, **kwargs) -> msticnb.nb.azsent.host.logon_session_rarity.LogonSessionsRarityResult
Calculate Logon sessions ordered by process rarity summary.

Inherited methods

check_table_exists

check_table_exists(self, table: str) -> bool
Check to see if the table exists in the provider.

check_valid_result_data

check_valid_result_data(self, attrib: str = None, silent: bool = False) -> bool
Check that the result is valid and `attrib` contains data.

get_methods

get_methods(self) -> Dict[str, Callable[[Any], Any]]
Return methods available for this class.

get_pivot_run

get_pivot_run(self, get_timespan: Callable[[], msticpy.common.timespan.TimeSpan])
Return Pivot-wrappable run function.

get_provider

get_provider(self, provider_name: str)
Return data provider for the specified name.

list_methods

list_methods(self) -> List[str]
Return list of methods with descriptions.

run_nb_func

run_nb_func(self, nb_func: Union[str, msticnb.notebooklet_func.NBFunc], **kwargs)
Run the notebooklet function and return the results.

run_nb_funcs

run_nb_funcs(self)
Run all notebooklet functions defined for the notebooklet.

Other Methods

add_nb_function

add_nb_function(nb_func: Union[str, msticnb.notebooklet_func.NBFunc], **kwargs)
Add a notebooklet function to the class.

all_options

all_options() -> List[str]
Return supported options for Notebooklet run function.

default_options

default_options() -> List[str]
Return default options for Notebooklet run function.

description

description() -> str
Return description of the Notebooklet.

entity_types

entity_types() -> List[str]
Entity types supported by the notebooklet.

get_help

get_help(fmt='html') -> str
Return HTML document for class.

get_settings

get_settings(print_settings=True) -> Optional[str]
Print or return metadata for class.

import_cell

import_cell()
Import the text of this module into a new cell.

keywords

keywords() -> List[str]
Return search keywords for Notebooklet.

list_options

list_options() -> str
Return options document for Notebooklet run function.

match_terms

match_terms(search_terms: str) -> Tuple[bool, int]
Search class definition for `search_terms`.

name

name() -> str
Return name of the Notebooklet.

print_options

print_options()
Print options for Notebooklet run function.

result

result [property] Return result of the most recent notebooklet run.

show_help

show_help()
Display Documentation for class.

silent

silent [property] Get the current instance setting for silent running.

<hr>

`run` function documentation

Calculate Logon sessions ordered by process rarity summary.

Parameters

value [str] Not used

data [Optional[pd.DataFrame], optional] Process event data.

timespan [TimeSpan] Not used

options [Optional[Iterable[str]], optional] List of options to use, by default None. A value of None means use default options. Options prefixed with "+" will be added to the default options. To see the list of available options type *help(cls)* where "cls" is the notebooklet class or an instance of this class.

Returns

LogonSessionsRarityResult LogonSessionsRarityResult.

Raises

MsticnbMissingParameterError If required parameters are missing

Default Options

None

Other Options

None

2.1.7 Notebooklet Class - IpAddressSummary

IP Address Summary Notebooklet class.

Queries and displays summary information about an IP address, including:

- Basic IP address properties
- IpAddress entity (and Host entity, if a host could be associated)
- WhoIs and Geo-location
- Azure activity and network data (optional)
- Office activity summary (optional)
- Threat intelligence reports
- Related alerts and hunting bookmarks

Default Options

- geoip: Get geo location information for IP address.
- alerts: Get any alerts listing the IP address.
- host_logons: Find any hosts with logons using this IP address as a source.
- related_accounts: Find any accounts using this IP address in AAD or host logs.
- device_info: Find any devices associated with this IP address.
- device_network: Find any devices communicating with this IP address.

Other Options

- bookmarks: Get any hunting bookmarks listing the IP address.
- heartbeat: Get the latest heartbeat record for for this IP address.
- az_net_if: Get the latest Azure network analytics interface data for this IP address.
- vmcomputer: Get the latest VMComputer record for this IP address.
- az_netflow: Get netflow information from AzureNetworkAnalytics table.
- passive_dns: Force fetching passive DNS data from a TI Provider even if IP is internal.
- az_activity: AAD sign-ins and Azure Activity logs.
- office_365: Office 365 activity.
- common_security: Get records from common security log.
- ti: Force get threat intelligence reports even for internal public IPs.

Display Sections

Azure Sign-ins and audit activity from IP Address

(only available for Azure)

Azure Azure NSG Flow Logs for IP

(only available for if Azure network analytics net flow enabled.) This is is a list of netflow events for the IP. Timeline by protocol is available in the `result.az_network_flows_timeline` property - Use `nblt.netflow_total_by_protocol()` method to view flow totals by protocol - Use `nblt.netflow_total_by_direction()` to view a timeline grouped by direction of flow

Office 365 operations summary from IP Address

(only available for Office 365)

Public IP data (GeoIP, ThreatIntel, Passive DNS, VPS membership)

Azure Sentinel alerts related to the IP

Use `nblt.browse_alerts()` to retrieve a list of alerts.

Azure Sentinel alerts related to the IP

Use `nblt.browse_alerts()` to retrieve a list of alerts.

IP Address summary

Retrieving data for IP Address Data and plots are stored in the result class returned by this function.

Azure Network Analytics Topology record for the IP

(only available for Azure VMs)

Common security log

The CommonSecurityLog contains log data from firewalls and network devices.

Defender device information

MS Defender device network and host information.

Network connections

MS Defender network connections to/from this IP address.

Azure Sentinel heartbeat record for the IP

(only available for IP addresses that belong to the subscription)

Host logons

List of hosts with logon attempts from this IP address.

Related accounts

List of accounts using the IP address.

Azure VMComputer record for the IP.

(only available for Azure VMs)

Summary of Azure NSG network flow data for this IP Address

(only available for if Azure network analytics net flow enabled.)

Results Class

IPSummary Results.

- ip_str : str
  The input IP address as a string.
- ip_address : Optional[Union[IPv4Address, IPv6Address]]

Ip Address Python object

- ip_entity : IpAddress
  IpAddress entity
- ip_origin : str
  "External" or "Internal"
- host_entities : Host
  Host entity or entities associated with IP Address
- ip_type : str
  IP address type - "Public", "Private", etc.
- geoip : Optional[Dict[str, Any]]
  Geo location information as a dictionary.
- location : Optional[GeoLocation]
  Location entity context object.
- whois : pd.DataFrame
  WhoIs information for IP Address
- whois_nets : pd.DataFrame
  List of networks definitions from WhoIs data
- heartbeat : pd.DataFrame
  Heartbeat record for IP Address or host
- az_network_if : pd.DataFrame
  Azure NSG analytics interface record, if available
- vmcomputer : pd.DataFrame
  VMComputer latest record
- az_network_flows : pd.DataFrame
  Azure NSG flows for IP, if available
- az_network_flows_timeline: Figure
  Azure NSG flows timeline, if data is available
- aad_signins : pd.DataFrame = None
  AAD signin activity
- azure_activity : pd.DataFrame = None
  Azure Activity log entries
- azure_activity_summary : pd.DataFrame = None
  Azure Activity (AAD and Az Activity) summarized view
- office_activity : pd.DataFrame = None
  Office 365 activity
- common_security : pd.DataFrame
  Common Security Log entries for source IP
- related_bookmarks : pd.DataFrame
  Bookmarks related to IP Address
- alert_timeline : Figure
  Timeline plot of alerts

- ti_results: pd.DataFrame
  Threat intel lookup results
- passive_dns: pd.DataFrame
  Passive DNS lookup results
- self.host_logons : pd.DataFrame
  Hosts with logons from this IP Address
- self.related_accounts : pd.DataFrame
  Accounts with activity related to this IP Address
- self.associated_hosts : pd.DataFrame
  Hosts using this IP Address
- self.device_info : pd.DataFrame
  Device info of hosts using this IP Address
- self.network_connections : pd.DataFrame = None
  Network connections to/from this IP on other devices

Methods

__init__

__init__(self, data_providers: Optional[<msticnb.data_providers.SingletonDecorator object at 0x0000023FAFA3A6A0>] = None, **kwargs)
Initialize a new instance of the notebooklet class.

browse_alerts

browse_alerts(self) -> msticpy.nbtools.nbwidgets.select_alert.SelectAlert
Return alert browser/viewer.

browse_ti_results

browse_ti_results(self)
Display Threat intel results.

display_alert_timeline

display_alert_timeline(self)
Display the alert timeline.

netflow_by_direction

netflow_by_direction(self) -> bokeh.plotting.figure.Figure
Display netflows grouped by direction.

netflow_by_protocol

netflow_by_protocol(self) -> bokeh.plotting.figure.Figure
Display netflows grouped by protocol.

netflow_total_by_protocol

netflow_total_by_protocol(self) -> bokeh.plotting.figure.Figure
Display netflows grouped by protocol.

run

run(self, value: Any = None, data: Optional[pandas.core.frame.DataFrame] = None, timespan: Optional[msticpy.common.timespan.TimeSpan] = None, options: Optional[Iterable[str]] = None, **kwargs) -> msticnb.nb.azsent.network.ip_summary.IpSummaryResult
Return IP Address activity summary.

Inherited methods

check_table_exists

check_table_exists(self, table: str) -> bool
Check to see if the table exists in the provider.

check_valid_result_data

check_valid_result_data(self, attrib: str = None, silent: bool = False) -> bool
Check that the result is valid and `attrib` contains data.

get_methods

get_methods(self) -> Dict[str, Callable[[Any], Any]]
Return methods available for this class.

get_pivot_run

get_pivot_run(self, get_timespan: Callable[[], msticpy.common.timespan.TimeSpan])
Return Pivot-wrappable run function.

get_provider

get_provider(self, provider_name: str)
Return data provider for the specified name.

list_methods

list_methods(self) -> List[str]
Return list of methods with descriptions.

run_nb_func

run_nb_func(self, nb_func: Union[str, msticnb.notebooklet_func.NBFunc], **kwargs)
Run the notebooklet function and return the results.

run_nb_funcs

run_nb_funcs(self)
Run all notebooklet functions defined for the notebooklet.

Other Methods

add_nb_function

add_nb_function(nb_func: Union[str, msticnb.notebooklet_func.NBFunc], **kwargs)
Add a notebooklet function to the class.

all_options

all_options() -> List[str]
Return supported options for Notebooklet run function.

default_options

default_options() -> List[str]
Return default options for Notebooklet run function.

description

description() -> str
Return description of the Notebooklet.

entity_types

entity_types() -> List[str]
Entity types supported by the notebooklet.

get_help

get_help(fmt='html') -> str
Return HTML document for class.

get_settings

get_settings(print_settings=True) -> Optional[str]
Print or return metadata for class.

import_cell

import_cell()
Import the text of this module into a new cell.

keywords

keywords() -> List[str]
Return search keywords for Notebooklet.

list_options

list_options() -> str
Return options document for Notebooklet run function.

match_terms

match_terms(search_terms: str) -> Tuple[bool, int]
Search class definition for `search_terms`.

name

name() -> str
Return name of the Notebooklet.

print_options

print_options()
Print options for Notebooklet run function.

result

result [property] Return result of the most recent notebooklet run.

show_help

show_help()
Display Documentation for class.

silent

silent [property] Get the current instance setting for silent running.

<hr>

`run` function documentation

Return IP Address activity summary.

value [str] IP Address - The key for searches

data [Optional[pd.DataFrame], optional] Not supported for this notebooklet.

timespan [TimeSpan] Timespan for queries

options [Optional[Iterable[str]], optional] List of options to use, by default None. A value of None means use default options. Options prefixed with "+" will be added to the default options. To see the list of available options type *help(cls)* where "cls" is the notebooklet class or an instance of this class.

IpSummaryResult Result object with attributes for each result type.

MsticnbMissingParameterError If required parameters are missing

- geoip: Get geo location information for IP address.
- alerts: Get any alerts listing the IP address.
- host_logons: Find any hosts with logons using this IP address as a source.
- related_accounts: Find any accounts using this IP address in AAD or host logs.
- device_info: Find any devices associated with this IP address.
- device_network: Find any devices communicating with this IP address.
- bookmarks: Get any hunting bookmarks listing the IP address.
- heartbeat: Get the latest heartbeat record for for this IP address.
- az_net_if: Get the latest Azure network analytics interface data for this IP address.
- vmcomputer: Get the latest VMComputer record for this IP address.
- az_netflow: Get netflow information from AzureNetworkAnalytics table.
- passive_dns: Force fetching passive DNS data from a TI Provider even if IP is internal.
- az_activity: AAD sign-ins and Azure Activity logs.
- office_365: Office 365 activity.

- common_security: Get records from common security log.
- ti: Force get threat intelligence reports even for internal public IPs.

2.1.8 Notebooklet Class - NetworkFlowSummary

Network Flow Summary Notebooklet class.

Queries network data and plots time lines for network

traffic to/from a host or IP address.

- Plot flows events by protocol and direction
- Plot flow count by protocol
- Display flow summary table
- Display flow summary by ASN
- Display results on map

Methods

- run: main method for notebooklet.
- select_asns: Open an interactive dialog to choose which ASNs to

investigate further.

- lookup_ti_for_asn_ips: For selected ASNs, lookup Threat Intelligence

data for the IPs belonging to those ASNs.

- show_selected_asn_map: Show IP address locations for selected IP

(including any threats highlighted)

Default Options

- plot_flows: Create plots of flows by protocol and direction.
- plot_flow_values: Plot flow county by protocol.
- flow_summary: Create a summarization of all flows and all flows grouped by ASN.
- resolve_host: Try to resolve the host name before other operations.

Other Options

- geo_map: Plot a map of all IP address locations in communication with the host (see the method below for plotting selected IPs only).

Display Sections

Host Network Summary

This shows a summary of network flows for this endpoint. Data and plots are stored in the result class returned by this function.

Map of geographic location of selected IPs communicating with host

Numbered circles indicate multiple items - click to expand these. Hovering over a location shows brief details, clicking on an IP location shows more detail. Location marker key - Blue = outbound - Purple = inbound - Green = Host - Red = Threats

Map of geographic location of all IPs communicating with host

Numbered circles indicate multiple items - click to expand these. Hovering over a location shows brief details, clicking on an IP location shows more detail. Location marker key - Blue = outbound - Purple = inbound - Green = Host

Flow Index.

List of flows grouped by source, dest, protocol and direction.

Flow Summary with ASN details.

Gets the ASN details from WhoIs. The data shows flows grouped by source and destination ASNs. All protocol types and all source IP addresses are grouped into lists for each ASN.

TI Lookup for IP Addresses in selected ASNs.

The remote IPs from each selected ASN are are searched for your selected Threat Intelligence providers. Check the results to see if there are indications of malicious activity associated with these IPs.

Timeline of network flows quantity.

Each protocol is plotted as a separate colored series. The vertical axis indicates the number for flows recorded for that time slot.

Timeline of network flows by direction.

I = inbound, O = outbound.

Timeline of network flows by protocol type.

Select the ASNs to process.

Choose any unusual looking ASNs that you want to examine. The remote IPs from each selected ASN will be sent to your selected Threat Intelligence providers to check if there are indications of malicious activity associated with these IPs. By default, the most infrequently accessed ASNs are selected.

Results Class

NetworkFlowResult

Network Flow Details Results.

Attributes

- host_entity : msticpy.datamodel.entities.Host
  The host entity object contains data about the host such as name, environment, operating system version, IP addresses and Azure VM details. Depending on the type of host, not all of this data may be populated.
- network_flows : pd.DataFrame
  The raw network flows recorded for this host.
- plot_flows_by_protocol : Figure
  Bokeh timeline plot of flow events by protocol.
- plot_flows_by_direction : Figure
  Bokeh timeline plot of flow events by direction (in/out).
- plot_flow_values : Figure
  Bokeh values plot of flow events by protocol.
- flow_index : pd.DataFrame
  Summarized DataFrame of flows
- flow_index_data : pd.DataFrame
  Raw summary data of flows.
- flow_summary : pd.DataFrame
  Summarized flows grouped by ASN
- ti_results : pd.DataFrame
  Threat Intelligence results for selected IP Addreses.
- geo_map : foliummap.FoliumMap
  Folium map showing locations of all IP Addresses.
- geo_map_selected : foliummap.FoliumMap
  Folium map showing locations of selected IP Addresses.

Methods

Instance Methods

__init__

__init__(self, data_providers: Optional[<msticnb.data_providers.SingletonDecorator object at 0x0000023FAFA3A6A0>] = None, **kwargs)
Intialize a new instance of the notebooklet class.

lookup_ti_for_asn_ips

lookup_ti_for_asn_ips(self)
Lookup Threat Intel data for IPs of selected ASNs.

run

run(self, value: Any = None, data: Optional[pandas.core.frame.DataFrame] = None, timespan: Optional[msticpy.common.timespan.TimeSpan] = None, options: Optional[Iterable[str]] = None, **kwargs) -> msticnb.nb.azsent.network.network_flow_summary.NetworkFlowResult
Return host summary data.

select_asns

select_asns(self)
Show interactive selector to choose which ASNs to process.

show_selected_asn_map

show_selected_asn_map(self) -> msticpy.nbtools.foliummap.FoliumMap
Display map of IP locations for selected ASNs.

Inherited methods

check_table_exists

check_table_exists(self, table: str) -> bool
Check to see if the table exists in the provider.

check_valid_result_data

check_valid_result_data(self, attrib: str = None, silent: bool = False) -> bool
Check that the result is valid and `attrib` contains data.

get_methods

get_methods(self) -> Dict[str, Callable[[Any], Any]]
Return methods available for this class.

get_pivot_run

get_pivot_run(self, get_timespan: Callable[[], msticpy.common.timespan.TimeSpan])
Return Pivot-wrappable run function.

get_provider

get_provider(self, provider_name: str)
Return data provider for the specified name.

list_methods

list_methods(self) -> List[str]
Return list of methods with descriptions.

run_nb_func

run_nb_func(self, nb_func: Union[str, msticnb.notebooklet_func.NBFunc], **kwargs)
Run the notebooklet function and return the results.

run_nb_funcs

run_nb_funcs(self)
Run all notebooklet functions defined for the notebooklet.

Other Methods

add_nb_function

add_nb_function(nb_func: Union[str, msticnb.notebooklet_func.NBFunc], **kwargs)
Add a notebooklet function to the class.

all_options

all_options() -> List[str]
Return supported options for Notebooklet run function.

default_options

default_options() -> List[str]
Return default options for Notebooklet run function.

description

description() -> str
Return description of the Notebooklet.

entity_types

entity_types() -> List[str]
Entity types supported by the notebooklet.

get_help

get_help(fmt='html') -> str
Return HTML document for class.

get_settings

get_settings(print_settings=True) -> Optional[str]
Print or return metadata for class.

import_cell

import_cell()
Import the text of this module into a new cell.

keywords

keywords() -> List[str]
Return search keywords for Notebooklet.

list_options

list_options() -> str
Return options document for Notebooklet run function.

match_terms

match_terms(search_terms: str) -> Tuple[bool, int]
Search class definition for `search_terms`.

name

name() -> str
Return name of the Notebooklet.

print_options

print_options()
Print options for Notebooklet run function.

result

result [property] Return result of the most recent notebooklet run.

show_help

show_help()
Display Documentation for class.

silent

silent [property] Get the current instance setting for silent running.

<hr>

`run` function documentation

Return host summary data.

Parameters

value [str] Host entity, hostname or host IP Address

data [Optional[pd.DataFrame], optional] Not used, by default None

timespan [TimeSpan] Timespan over which operations such as queries will be performed, by default None. This can be a TimeStamp object or another object that has valid *start*, *end*, or *period* attributes.

options [Optional[Iterable[str]], optional] List of options to use, by default None A value of None means use default options. Options prefixed with "+" will be added to the default options. To see the list of available options type *help(cls)* where "cls" is the notebooklet class or an instance of this class.

Other Parameters

start [Union[datetime, datelike-string]] Alternative to specifying timespan parameter.

end [Union[datetime, datelike-string]] Alternative to specifying timespan parameter.

Returns

HostNetworkResult Result object with attributes for each result type.

Raises

MsticnbMissingParameterError If required parameters are missing

Default Options

- plot_flows: Create plots of flows by protocol and direction.
- plot_flow_values: Plot flow county by protocol.
- flow_summary: Create a summarization of all flows and all flows grouped by ASN.
- resolve_host: Try to resolve the host name before other operations.

Other Options

- geo_map: Plot a map of all IP address locations in communication with the host (see the method below for plotting selected IPs only).

2.1.9 Notebooklet Class - WinHostEvents

Windows host Security Events Notebooklet class.

Queries and displays Windows Security Events including:

- All security events summary
- Extracting and displaying account management events
- Account management event timeline
- Optionally parsing packed event data into DataFrame columns

Process (4688) and Account Logon (4624, 4625) are not included

in the event types processed by this module.

Default Options

- event_pivot: Display a summary of all event types.
- acct_events: Display a summary and timeline of account management events.

Other Options

- expand_events: parses the XML EventData column into separate DataFrame columns. This can be very expensive with a large event set. We recommend using the expand_events() method to select a specific subset of events to process.

Display Sections

Host Security Events Summary

This shows a summary of security events for the host. These are grouped by EventID and Account. Data and plots are stored in the result class returned by this function.

Summary of Account Management Events on host

This shows the subset of events related to account management, for example, creation/deletion of accounts, changes to group membership, etc. Yellow highlights indicate account with highest event count.

Timeline of Account Management Events on host

Summary of Security Events on host

This is a summary of Security events for the host (excluding process creation and account logon - 4688, 4624, 4625). Yellow highlights indicate account with highest event count for an EventID.

Parsing eventdata into columns

This may take some time to complete for large numbers of events. Since event types have different schema, some of the columns will not be populated for certain Event IDs and will show as `NaN`.

Results Class

WinHostEventsResult

Windows Host Security Events Results.

Attributes

- all_events : pd.DataFrame
  DataFrame of all raw events retrieved.
- event_pivot : pd.DataFrame
  DataFrame that is a pivot table of event ID vs. Account
- account_events : pd.DataFrame
  DataFrame containing a subset of account management events such as account and group modification.
- acct_pivot : pd.DataFrame
  DataFrame that is a pivot table of event ID vs. Account of account management events
- account_timeline : Union[Figure, LayoutDOM]
  Bokeh plot figure or Layout showing the account events on an interactive timeline.
- expanded_events : pd.DataFrame
  If `expand_events` option is specified, this will contain the parsed/expanded EventData as individual columns.

Methods

Instance Methods

__init__

__init__(self, data_providers: Optional[<msticnb.data_providers.SingletonDecorator object at 0x0000023FAFA3A6A0>] = None, **kwargs)
Initialize a new instance of the notebooklet class.

expand_events

expand_events(self, event_ids: Union[int, Iterable[int], NoneType] = None) -> pandas.core.frame.DataFrame
Expand `EventData` for `event_ids` into separate columns.

run

run(self, value: Any = None, data: Optional[pandas.core.frame.DataFrame] = None, timespan: Optional[msticpy.common.timespan.TimeSpan] = None, options: Optional[Iterable[str]] = None, **kwargs) -> msticnb.nb.azsent.host.win_host_events.WinHostEventsResult
Return Windows Security Event summary.

Inherited methods

check_table_exists

check_table_exists(self, table: str) -> bool
Check to see if the table exists in the provider.

check_valid_result_data

check_valid_result_data(self, attrib: str = None, silent: bool = False) -> bool
Check that the result is valid and `attrib` contains data.

get_methods

get_methods(self) -> Dict[str, Callable[[Any], Any]]
Return methods available for this class.

get_pivot_run

get_pivot_run(self, get_timespan: Callable[[], msticpy.common.timespan.TimeSpan])
Return Pivot-wrappable run function.

get_provider

get_provider(self, provider_name: str)
Return data provider for the specified name.

list_methods

list_methods(self) -> List[str]
Return list of methods with descriptions.

run_nb_func

run_nb_func(self, nb_func: Union[str, msticnb.notebooklet_func.NBFunc], **kwargs)
Run the notebooklet function and return the results.

run_nb_funcs

run_nb_funcs(self)
Run all notebooklet functions defined for the notebooklet.

Other Methods

add_nb_function

add_nb_function(nb_func: Union[str, msticnb.notebooklet_func.NBFunc], **kwargs)
Add a notebooklet function to the class.

all_options

all_options() -> List[str]
Return supported options for Notebooklet run function.

default_options

default_options() -> List[str]
Return default options for Notebooklet run function.

description

description() -> str
Return description of the Notebooklet.

entity_types

entity_types() -> List[str]
Entity types supported by the notebooklet.

get_help

get_help(fmt='html') -> str
Return HTML document for class.

get_settings

get_settings(print_settings=True) -> Optional[str]
Print or return metadata for class.

import_cell

import_cell()
Import the text of this module into a new cell.

keywords

keywords() -> List[str]
Return search keywords for Notebooklet.

list_options

list_options() -> str
Return options document for Notebooklet run function.

match_terms

match_terms(search_terms: str) -> Tuple[bool, int]
Search class definition for `search_terms`.

name

name() -> str
Return name of the Notebooklet.

print_options

print_options()
Print options for Notebooklet run function.

result

result [property] Return result of the most recent notebooklet run.

show_help

show_help()
Display Documentation for class.

silent

silent [property] Get the current instance setting for silent running.

<hr>

`run` function documentation

Return Windows Security Event summary.

Parameters

value [str] Host name

data [Optional[pd.DataFrame], optional] Not used, by default None

timespan [TimeSpan] Timespan over which operations such as queries will be performed, by default None. This can be a TimeStamp object or another object that has valid *start*, *end*, or *period* attributes.

options [Optional[Iterable[str]], optional] List of options to use, by default None. A value of None means use default options. Options prefixed with "+" will be added to the default options. To see the list of available options type *help(cls)* where "cls" is the notebooklet class or an instance of this class.

Other Parameters

start [Union[datetime, datelike-string]] Alternative to specifying timespan parameter.

end [Union[datetime, datelike-string]] Alternative to specifying timespan parameter.

Returns

HostSummaryResult Result object with attributes for each result type.

Raises

MsticnbMissingParameterError If required parameters are missing

Default Options

- event_pivot: Display a summary of all event types.
- acct_events: Display a summary and timeline of account management events.

Other Options

- expand_events: parses the XML EventData column into separate DataFrame columns. This can be very expensive with a large event set. We recommend using the expand_events() method to select a specific subset of events to process.

CHAPTER 3

Creating Notebooklets

3.1 Creating Notebooklets

Most of the process of creating a notebook is documented in the nb_template module. You can use this as a starting point for creating a notebooklet.

Notebooklets have two components:

- A python module containing the code that does all of the processing work that you'd normally write directly into notebook cells.
- A YAML file that contains configuration, documentation and text content that you want to display as part of your notebooklet's output.

Custom notebooklets must be in a package of their own (although you can have multiple notebooklets in the same package) so also require an `__init__.py` in the same folder.

Notebooklets are loaded by calling `nb.discover_modules()` function and specifying the path to the notebooklets package with the `nb_path` parameter. (see *`discover_modules`*)

3.1.1 Notebooklet module

The notebooklet module has three main sections:

- **Result class definition**: This defines the attributes and descriptions of the data that you want to return from the notebooklet.
- **Notebooklet class definition**: This is the entry point for running the notebooklet. At minimum it should be a class derived from Notebooklet that implements a *run* method and returns your result class.
- **Functions**: These do most of the work of the notebooklet and usually the code that is copied from or adapted from the original notebook.

Having the latter section is optional. You can choose to implement this functionality in instance methods of the notebooklet class.

However, there are advantages to keeping these as separate functions outside the class. It means that all the data used in the functions has to be passed around as parameters and return values. This can improve the clarity of the code and reduce errors due to some dependency on some mysterious global state.

If the user of your notebooklet wants to import the module's code into a notebook to read and possibly adapt it, having standalone functions will make it easier from them understand and work with the code.

Results Class

This is derived from the `NotebookletResult` It is also an attrs class so needs to be decorated with the @attr decorator.

```
@attr.s(auto_attribs=True)
class TemplateResult(NotebookletResult):
    """
    Template Results.

    Attributes
    ----------
    all_events : pd.DataFrame
        DataFrame of all raw events retrieved.
    plot : bokeh.models.LayoutDOM
        Bokeh plot figure showing the account events on an
        interactive timeline.
    additional_info: dict
        Additional information for my notebooklet.

    """

    description: str = "Windows Host Security Events"

    # Add attributes as needed here.
    # Make sure they are documented in the Attributes section
    # above.
    all_events: pd.DataFrame = None
    plot: Figure = None
    additional_info: Optional[dict] = None
```

The class is just a collection of attributes containing results that you want to return to the user. It is a good idea to add type hints that define what data type each attribute contains. Adding documentation for each attribute is important. This not only helps when reading the code or using the Python help() function but it is also used to automatically generate titles and descriptive text when you display the results class.

The Notebooklet class

The notebooklet class is the main engine behind a notebooklet. It is derived from *Notebooklet*

```
class TemplateNB(Notebooklet):
    """
    Template Notebooklet class.

    Detailed description of things this notebooklet does:

    - Fetches all events from XYZ
    - Plots interesting stuff
```

(continues on next page)

(continued from previous page)

```
    - Returns extended metadata about the thing

    Document the options that the Notebooklet takes, if any,
    Use these control which parts of the notebooklet get run.

    """
    # assign metadata from YAML to class variable
    metadata = _CLS_METADATA
    __doc__ = nb_metadata.update_class_doc(__doc__, metadata)
    _cell_docs = _CELL_DOCS
```

The first section of the the class definition contains the docstring. This documentation is used by the notebooklet browser and the show_help() function to provide extended user-friendly help.

The first three lines of code handle assigning metadata and documentation data from the notebooklet YAML file (see below) so that the notebooklet code can access it.

Warning: Do not change these lines unless you know what you are doing.

The run method

Notebooklet.run

The next section is the all-important `run` method. This method is the main entry point to the notebooklet and controls the flow of most of the logic. You can add other methods to do subsequent tasks but you should always implement a run method.

```
# @set_text decorator will display the title and text every time
# this method is run.
# The key value refers to an entry in the `output` section of
# the notebooklet yaml file.
@set_text(docs=_CELL_DOCS, key="run")
def run(
    self,
    value: Any = None,
    data: Optional[pd.DataFrame] = None,
    timespan: Optional[TimeSpan] = None,
    options: Optional[Iterable[str]] = None,
    **kwargs,
) -> TemplateResult:
    """
    Return XYZ summary.

    Parameters
    ----------
    value : str
        Host name - The key for searches - e.g. host, account, IPaddress
    data : Optional[pd.DataFrame], optional
        Alternatively use a DataFrame as input.
    timespan : TimeSpan
        Timespan for queries
    options : Optional[Iterable[str]], optional
        List of options to use, by default None.
        A value of None means use default options.
```

(continues on next page)

(continued from previous page)

```
        Options prefixed with "+" will be added to the default options.
        To see the list of available options type `help(cls)` where
        "cls" is the notebooklet class or an instance of this class.

    Returns
    -------
    TemplateResult
        Result object with attributes for each result type.

    Raises
    ------
    MsticnbMissingParameterError
        If required parameters are missing

    """
```

Most of this is class documentation - again this is used in the browser and user help so you should document this as shown. Usually you can just copy and paste this example and edit the text to suit your needs - for example, changing the description `value` if you are expecting an IP address.

Do not rename or add to these explicit parameters since they are referenced by the base class. If you want additional parameters you can supply them as keyword arguments and extract them from kwargs. Be sure to document any keyword arguments that you require.

The set_text decorator

The `@set_text` decorator requires some explanation. This decorator gives you the ability to output display text every time `run()` is called. It references the _CELL_DOCS dictionary, which is read from the YAML metadata file, and specifies a key which is used to look up the exact section from the file to use.

You can optionally add explicit title and text as parameters to `set_text` using the `title`, `text` and `hd_level` parameters. This is documented here *set_text*

The set_text decorator does not display any text if you run the notebooklet with `silent=True` parameter.

The run method body

```
# This line use logic in the superclass to populate options
# (including default options) into this class.
super().run(
    value=value, data=data, timespan=timespan, options=options, **kwargs
)
```

Calling the base class `run` method from your implementation is important. This does things like handle options and convert and normalize the timespan parameter.

The next section validates any input parameters that you require and creates a results class to store your output data. Assigning the description and the timespan being used to the results object is very helpful when you need to refer back to the result or possibly make additional ad hoc queries afterwards.

```
if not value:
    raise MsticnbMissingParameterError("value")
if not timespan:
    raise MsticnbMissingParameterError("timespan.")
```

(continues on next page)

(continued from previous page)

```
# Create a result class
result = TemplateResult()
result.description = self.metadata.description
result.timespan = timespan
```

The remainder of the run method is just about the logic of what you want to execute and in what order.

Note: be sure to assign your results class to `self._last_result`. This will expose the result class as a `result` property of your notebooklet instance and allow other methods in your class to reference it.

```
# You might want to always do some tasks irrespective of
# options sent
all_events_df = _get_all_events(
    self.query_provider, host_name=value, timespan=timespan
)
result.all_events = all_events_df

if "plot_events" in self.options:
    result.plot = _display_event_timeline(acct_event_data=all_events_df)

if "get_metadata" in self.options:
    result.additional_info = _get_metadata(host_name=value, timespan=timespan)

# Assign the result to the _last_result attribute
# so that you can get to it without having to re-run the operation
self._last_result = result  # pylint: disable=attribute-defined-outside-init

return self._last_result
```

You can call additional methods unconditionally or use the option logic to allow users to add additional operations or skip ones that they are not interested in. The available and default options for your notebooklet are defined in the notebooklet YAML file.

If you call run() without specifying the options parameter, the defaults will be used. You can specify a custom set of options as a list of option names (strings).

`options=["opt1", "opt2", "opt4"]`

You can also specify an incremental list. For example:

- `options=["+option_a"]` will add "option_a" to the list of default options.
- `options=["+option_a", "-option_b"]` will add "option_a" and remove "option_b" from the defaults.

Note: You cannot mix the explicit options with the incremental options syntax.

Be sure to assign the output from the called functions to the relevant attributes of your result and return the result at the end.

Additional notebooklet methods

Often you will not want to or not be able to execute additional functionality within the run command. You may require the user to choose an option before starting a second step or you may want to provide some kind of data browsing capability that is interactive and needs to the run method to have completed.

You can do this by adding methods to your notebooklet class. Any public methods you create will be added to the auto-documentation of the notebooklet.

This is an example method. Note that if you depend on the result being populated, you should check this and issue a warning if it is not (as shown).

```
def run_additional_operation(
    self, event_ids: Optional[Union[int, Iterable[int]]] = None
) -> pd.DataFrame:
    """
    Addition method.

    Parameters
    ----------
    event_ids : Optional[Union[int, Iterable[int]]], optional
        Single or interable of event IDs (ints).

    Returns
    -------
    pd.DataFrame
        Results with expanded columns.

    """
    # Include this to check the "run()" has happened before this method
    # can be run
    if (
        not self._last_result or self._last_result.all_events is None
    ):  # type: ignore
        print(
            "Please use 'run()' to fetch the data before using this method.",
            "\nThen call 'expand_events()'",
        )
        return None
    # Print a status message - this will not be displayed if
    # the user has set the global "verbose" option to False.
    nb_print("We maybe about to wait some time")

    nb_markdown("Print some message that always displays", "blue, bold")
    return _do_additional_thing(
        evt_df=self._last_result.all_events,  # type: ignore
        event_ids=event_ids,
    )
    # Note you can also assign new items to the result class in
    # self._last_result and return the updated result class.
```

One thing to note here is the use of *nb_markdown* and *nb_print* (there is also an *nb_display* function). These are simple wrappers around IPython.display.markdown(), Python print() and IPython.display.display(). These functions honor the `silent` parameter. This can be supplied to the notebooklet `__init__` method (when creating an instance of the class) or the `run` method. If silent is True then these functions do not display any output. You are free to use whatever output functions you choose but the notebooklet may produce unexpected output if the user has set the silent option to True.

Note: You can access `self.silent` to query the current setting. You can also set the silent option globally by using `nb.set_opt("silent", True)` (see *set_opt*)

Worker Functions

To keep the notebooklet class simple, most of the work done by the notebooklet is usually coded in separate module functions. These are usually declares as private functions by prefixing with "_"

This simple function executes a query and returns the results. The query provider, hostname and timespan are supplied in the call from the notebooklet run method.

```
def _get_all_events(qry_prov, host_name, timespan):
    # Tell the user that you're fetching data
    # (doesn't display if nb.set_opt("silent", True))
    nb_data_wait("SecurityEvent")
    return qry_prov.WindowsSecurity.list_host_events(
        timespan,
        host_name=host_name,
        add_query_items="| where EventID != 4688 and EventID != 4624",
    )
```

nb_data_wait just outputs a standard message telling the user that data is being retrieved.

This is another example showing the use of the `@set_text` decorator. The output from this will be displayed as the plot is shown. The plot layout object is returned to the notebooklet class and added to the results class (shown earlier).

```
@set_text(docs=_CELL_DOCS, key="display_event_timeline")
def _display_event_timeline(acct_event_data):
    # Plot events on a timeline

    # Note the nbdisplay function is a wrapper around IPython.display()
    # However, it honors the "silent" option (global or per-notebooklet)
    # which allows you to suppress output while running.
    return nbdisplay.display_timeline(
        data=acct_event_data,
        group_by="EventID",
        source_columns=["Activity", "Account"],
        legend="right",
    )
```

3.1.2 Notebook YAML file

The notebooklet YAML file should have the same name as the Python module but with a "yaml" or "yml" extension.

There are two main sections: `metadata` and `output`.

```
metadata:
    name: TemplateNB
    description: Template YAML for Notebooklet
    default_options:
        - all_events: Gets all events about blah
        - plot_events:
            Display and summary and timeline of events.
    other_options:
```

(continues on next page)

(continued from previous page)

```
        - get_metadata: fetches additional metadata about the entity
    keywords:
        - host
        - computer
        - heartbeat
        - windows
        - account
    entity_types:
        - host
    req_providers:
        - AzureSentinel|LocalData
        - tilookup
```

The metadata section defines runtime parameters for the notebooklet. These include:

- the notebooklet display name
- the notebooklet description
- the default options (a list of key/value pairs of option name and description)
- other options available
- keywords (used in searching for the notebooklet
- entity types - mainly informational so that a user can find all notebooklets that deal with hosts, IP addresses, etc.
- req_providers - this is a list of data providers required for the notebooklet to run. You can provide alternates (as shown), which means that if one of the providers is available the notebooklet will load successfully.

```
output:
    run:
        title: Title for the run method (main title)
        hd_level: 1
        text:
        Write your introductory text here

        Data and plots are stored in the result class returned by this function.

        If you use **markdown** syntax in this block add the following
        to use markdown processing.
        md: True
    display_event_timeline:
        title: Display the timeline.
        text: '
        This may take some time to complete for large numbers of events.

        It will do:
        - Item one
        - Item two

        Since some groups will be undefined these can show up as `NaN`.

        Note: use a quoted string if you want to include yaml reserved chars
        such as ":"
        '
        md: True
```

The output section defines the display text for the `@set_text` decorator function used in the notebooklet module. The key for each section under output must match the value for the `key` parameter in the call to `set_text`.

Each section has the following sub-keys:

- title: the title to display (by default as HTML h2 or Markdown "##")
- hd_level: (1-4) to override the default heading level
- text: the body text to display. This will display as plain text by default
- md: set to True to process the "text" value as Markdown.

CHAPTER 4

API

4.1 Core modules and classes

4.1.1 Submodules

msticnb.class_doc	Functions to create documentation from notebooklets classes.
msticnb.common	Common definitions and classes.
msticnb.data_providers	Data Providers class and init function.
msticnb.nb_browser	Jupyter Browser for Notebooklets.
msticnb.notebooklet	Notebooklet base classes.
msticnb.options	Notebooklets global options.
msticnb.read_modules	read_modules - handles reading notebooklets modules.

4.1.2 msticnb.class_doc module

Functions to create documentation from notebooklets classes.

msticnb.class_doc.**get_class_doc**(*doc_cls: type*, *fmt: str = 'html'*) → str
Create HTML documentation for the notebooklet class.

Parameters

- **doc_cls** (*type*) – The class to document
- **fmt** (*str*) – Format = "html" or "md", by default "html"

Returns HTML documentation for the class

Return type str

Raises TypeError – If the class is not a subclass of Notebooklet.

4.1.3 msticnb.common module

Common definitions and classes.

exception msticnb.common.**MsticnbDataProviderError**
Bases: *msticnb.common.MsticnbError*

DataProvider Error.

args

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception msticnb.common.**MsticnbError**
Bases: Exception

Generic exception class for Notebooklets.

args

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception msticnb.common.**MsticnbMissingParameterError**(**args*)
Bases: *msticnb.common.MsticnbError*

Parameter Error.

Exception for missing parameter.

Parameters **args** (*str*) – First arg is the name or names of the parameters.

args

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class msticnb.common.**NBContainer**
Bases: object

Container for Notebooklet classes.

iter_classes() → Iterable[Tuple[str, Any]]
Return iterator through all notebooklet classes.

msticnb.common.**add_result**(*result: Any, attr_name: Union[str, List[str]]*)
Decorate func to add return value(s) to *result*.

Parameters

- **result** (*Any*) – Object that will have result attributes set.
- **attr_name** (*str or List[str]*) – Name of return attribute to set on *result*

Returns Wrapped function

Return type Callable[*args, **kwargs]

msticnb.common.**check_mp_version**(*required_version: str*) → bool
Returns true if the installed version is >= *required_version*.

msticnb.common.**df_has_data**(*data*) → bool
Return True if *data* DataFrame has data.

msticnb.common.**mp_version**()
Return currently-loaded msticpy version.

msticnb.common.**nb_data_wait**(*source: str*)
Print Getting data message.

Parameters **source** (*str*) – The data source.

msticnb.common.**nb_debug**(**args*)
Print debug args.

msticnb.common.**nb_display**(**args*, ***kwargs*)
Ipython display function wrapper.

msticnb.common.**nb_markdown**(**args*, ***kwargs*)
Display Markdown/HTML text.

msticnb.common.**nb_print**(**args*, ***kwargs*)
Print output but suppress if "silent".

Parameters **mssg** (*Any*) – The item/message to show

msticnb.common.**nb_warn**(**args*, ***kwargs*)
Display Markdown/HTML warning text.

msticnb.common.**set_text**(*title: Optional[str] = None*, *hd_level: int = 2*, *text: Optional[str] = None*, *md: bool = False*, *docs: Dict[str, Any] = None*, *key: str = None*)
Decorate function to print title/text before execution.

Parameters

- **title** (*Optional[str], optional*) – Title text to print, by default None
- **hd_level** (*int*) – Heading level (1-4), by default 2
- **text** (*Optional[str], optional*) – Text to print, by default None
- **md** (*bool, optional*) – Treat *text* as markdown, by default False
- **docs** (*Dict[str, Any]*) – Dictionary of cell documentation indexed by *key*
- **key** (*str*) – Item to use from *docs* dictionary.

Returns Wrapped function

Return type Callable[*args, **kwargs]

msticnb.common.**show_bokeh**(*plot*)
Display bokeh plot, resetting output.

4.1.4 msticnb.data_providers module

Data Providers class and init function.

class msticnb.data_providers.**ProviderDefn**(*prov_class*, *connect_reqd*, *get_config*)
Bases: tuple

Create new instance of ProviderDefn(prov_class, connect_reqd, get_config)

connect_reqd
Alias for field number 1

count()
Return number of occurrences of value.

get_config
Alias for field number 2

index()
Return first index of value.

Raises ValueError if the value is not present.

prov_class
Alias for field number 0

class msticnb.data_providers.**SingletonDecorator**(*wrapped_cls*)
Bases: object

Singleton decorator class.

Notes

Using this decorator on a class enforces the following behavior: - First instantiation of class will work as normal - Subsequent attempts with the same set/values of kwargs

will just return the original class

- Instantiation of the class with a different set of kwargs will instantiate a new class.
- The class method *current()* will always return the last instance of the class.

Instantiate the class wrapper.

current()
Return the current instance of the wrapped class.

msticnb.data_providers.**init**(*query_provider: str = 'AzureSentinel'*, *providers: Optional[List[str]] = None*, ***kwargs*)
Instantiate an instance of DataProviders.

Parameters

- **query_provider** (*str, optional*) – DataEnvironment name of the primary query provider. By default, "AzureSentinel". You can add addtional query providers by including them in the *providers* list.
- **providers** (*Optional[List[str]], optional*) – A list of provider names, by default None

Other Parameters **kwargs** – You can pass parameters to individual providers using the following notation: *ProviderName_param_name="param_value" Where 'ProviderName* is the name of the data provider, *param_name* is the parameter name expected by the provider and *param_value* is the value to assign to *param_name*. *param_value* can be any type.

Depending on the provider, these parameters (with the prefix stripped) are sent to either the constructor or *connect* method.

Notes

To see a list of currently supported providers call: *DataProviders.list_providers()*

4.1.5 msticnb.nb_browser module

Jupyter Browser for Notebooklets.

class msticnb.nb_browser.**NBBrowser**
Bases: object

Interactive browser/viewer for Notebooklets.

Initialize and Display Notebooklet Browser.

display()
Display the widget.

4.1.6 msticnb.nb_metadata module

Notebooklet base classes.

class msticnb.nb_metadata.**NBMetadata**(*name: str = 'Unnamed', mod_name: str = '', description: str = '', default_options: List[Union[str, Dict[KT, VT]]] = NOTHING, other_options: List[Union[str, Dict[KT, VT]]] = NOTHING, inputs: List[str] = ['value'], entity_types: List[str] = NOTHING, keywords: List[str] = NOTHING, req_providers: List[str] = NOTHING*)
Bases: object

Notebooklet metadata class.

Method generated by attrs for class NBMetadata.

all_options
Return combination of default and other options.

get_options(*option_set: str = 'all'*) → List[Tuple[str, Optional[str]]]
Return list of options and descriptions.

Parameters **option_set** (*str, optional*) – The subset of options to return, by default "all" Other values are "default" and "other"

Returns A list of tuples of option name and description.

Return type List[Tuple[str, Optional[str]]]

options_doc
Return list of options and documentation.

search_terms
Return set of search terms for the object.

msticnb.nb_metadata.**read_mod_metadata**(*mod_path: str*, *module_name*) → Tuple[msticnb.nb_metadata.NBMetadata, Dict[str, Any]]
Read notebooklet metadata from yaml file.

Parameters

- **mod_path** (*str*) – The fully-qualified (dotted) module name
- **module_name** (*str*) – The full module name.

Returns A tuple of the metadata class and the documentation dictionary

Return type Tuple[*NBMetadata*, Dict[str, Any]]

msticnb.nb_metadata.**update_class_doc**(*cls_doc: str*, *cls_metadata: msticnb.nb_metadata.NBMetadata*)
Append the options documentation to the *cls_doc*.

4.1.7 msticnb.notebooklet module

Notebooklet base classes.

class msticnb.notebooklet.**Notebooklet**(*data_providers: Optional[msticnb.data_providers.DataProviders] = None*, ***kwargs*)

Bases: abc.ABC

Base class for Notebooklets.

Intialize a new instance of the notebooklet class.

Parameters **data_providers** (*DataProviders, Optional*) – Optional DataProviders instance to query data. Most classes require this.

Raises MsticnbDataProviderError – If DataProviders has not been initialized. If required providers are specified by the notebooklet but are not available.

classmethod all_options() → List[str]
Return supported options for Notebooklet run function.

Returns Supported options.

Return type List[str]

check_table_exists(*table: str*) → bool
Check to see if the table exists in the provider.

Parameters **table** (*str*) – Table name

Returns True if the table exists, otherwise False.

Return type bool

check_valid_result_data(*attrib: str = None*, *silent: bool = False*) → bool
Check that the result is valid and *attrib* contains data.

Parameters

- **attrib** (*str*) – Name of the attribute to check, if None this function only checks for a valid _last_result.
- **silent** (*bool*) – If True, suppress output.

Returns Returns True if valid data is available, else False.

Return type bool

classmethod default_options() → List[str]
Return default options for Notebooklet run function.

Returns Supported options.

Return type List[str]

classmethod description() → str
Return description of the Notebooklet.

Returns Description

Return type str

classmethod entity_types() → List[str]
Entity types supported by the notebooklet.

Returns Entity names

Return type List[str]

classmethod get_help(*fmt='html'*) → str
Return HTML document for class.

get_methods() → Dict[str, Callable[[Any], Any]]
Return methods available for this class.

get_pivot_run(*get_timespan: Callable[[], msticpy.common.timespan.TimeSpan]*)
Return Pivot-wrappable run function.

get_provider(*provider_name: str*)
Return data provider for the specified name.

Parameters **provider_name** (*str*) – Name of the provider

Returns Provider instance.

Return type Any

Raises MsticnbDataProviderError – If provider is not found.

classmethod get_settings(*print_settings=True*) → Optional[str]
Print or return metadata for class.

Parameters **print_settings** (*bool, optional*) – Print to standard, by default True or return the str formatted content.

Returns If *print_settings* is True, returns None. If False, returns LF-delimited string of metadata settings.

Return type Optional[str]

Notes

Use *metadata* attribute to retrieve the metadata directly.

classmethod import_cell()
Import the text of this module into a new cell.

classmethod keywords() → List[str]
Return search keywords for Notebooklet.

Returns Keywords

Return type List[str]

list_methods() → List[str]
Return list of methods with descriptions.

classmethod list_options() → str
Return options document for Notebooklet run function.

Returns Supported options.

Return type List[str]

classmethod match_terms(*search_terms: str*) → Tuple[bool, int]
Search class definition for *search_terms*.

Parameters **search_terms** (*str*) – One or more search terms, separated by spaces or commas. Terms can be simple strings or regular expressions.

Returns Returns a tuple of bool (True if all terms match) and int (count of matched terms)

Return type Tuple[bool, int]

metadata = NBMetadata(name='Notebooklet', mod_name='', description='Base class', defau

module_path = ''

classmethod name() → str
Return name of the Notebooklet.

Returns Name

Return type str

classmethod print_options()
Print options for Notebooklet run function.

result
Return result of the most recent notebooklet run.

Returns Notebooklet result class or None if nothing has been run.

Return type Optional[*NotebookletResult*]

run(*value: Any = None*, *data: Optional[pandas.core.frame.DataFrame] = None*, *timespan: Optional[msticpy.common.timespan.TimeSpan] = None*, *options: Optional[Iterable[str]] = None*, ***kwargs*) → msticnb.notebooklet_result.NotebookletResult
Notebooklet abstract base class.

Parameters

- **value** (*Any, optional*) – value to process, by default None
- **data** (*Optional[pd.DataFrame], optional*) – Input data to process, by default None
- **timespan** (*Optional[TimeSpan, Any], optional*) – Timespan over which operations such as queries will be performed, by default None. This can be a TimeStamp object or another object that has valid *start*, *end*, or *period* attributes.
- **options** (*Optional[Iterable[str]], optional*) – List of options to use, by default None A value of None means use default options. Options prefixed with "+" will be added to the default options. Options prefixed with "-" will be removed from the default options. To see the list of available options type *help(cls)* where "cls" is the notebooklet class or an instance of this class.

Other Parameters

- **start** (*Union[datetime, datelike-string]*) – Alternative to specifying timespan parameter.
- **end** (*Union[datetime, datelike-string]*) – Alternative to specifying timespan parameter.

Returns Result class from the notebooklet

Return type *NotebookletResult*

classmethod show_help()
Display Documentation for class.

silent
Get the current instance setting for silent running.

Returns Silent running is enabled.

Return type Optional[bool]

4.1.8 msticnb.notebooklet_result module

Notebooklet Result base classes.

class msticnb.notebooklet_result.**NotebookletResult**(*description: Optional[str] = None, timespan: Optional[msticpy.common.timespan.TimeSpan] = None, notebooklet: Optional[Any] = None*)

Bases: *msticnb.data_viewers.DFViewer*

Base result class.

Create new Notebooklet result instance.

Parameters

- **description** (*Optional[str], optional*) – Result description, by default None
- **timespan** (*Optional[TimeSpan], optional*) – TimeSpan for the results, by default None
- **notebooklet** (*Optional[Notebooklet], optional*) – Originating notebooklet, by default None

data_properties(*empty: bool = False*) → List[str]
Return list of attributes with populated data.

prop_doc(*name*) → Tuple[str, str]
Get the property documentation for the property.

properties
Return names of all properties.

view_events(*summary_cols: List[str] = None, attrib: Optional[str] = None, data: Optional[pandas.core.frame.DataFrame] = None, **kwargs*) → msticpy.nbtools.nbwidgets.select_item.SelectItem
Return simple data view for DataFrame/result attribute.

Parameters

- **summary_cols** (*List[str], optional*) – [description]
- **attrib** (*Optional[str], optional*) – [description], by default None
- **data** (*Optional[pd.DataFrame], optional*) – [description], by default None
- **kwargs** – Additional keyword arguments passed to the SelectItem widget.

Returns Browser for events in DataFrame.

Return type SelectItem

Raises

- `AttributeError` – Attribute name not in results class.
- `TypeError` – Input data or attribute is not a DataFrame
- `MsticnbMissingParameterError` – One of *data* or *attrib* parameters must be supplied
- `KeyError` – Summary column name specified that isn't in the DataFrame

vis_properties() → List[str]
Return list of properties with visualizations.

4.1.9 msticnb.options module

Notebooklets global options.

Available options are: [name, type (default value), description]

- *verbose*: bool (True) - Show progress messages.
- *debug*: bool (False) - Turn on debug output.
- *show_sample_results*: bool (True) - Display sample of results as they are produced.
- *silent*: bool (False) - Execute notebooklets with no output.

msticnb.options.**current**()
Show current settings.

msticnb.options.**get_opt**(*option: str*) → Any
Get the named option.

Parameters **option** (*str*) – Option name.

Returns Option value

Return type Any

Raises KeyError – An invalid option name was supplied.

msticnb.options.**set_opt**(*option: str*, *value: Any*)
Set the named option.

Parameters

- **option** (*str*) – Option name.
- **value** (*Any*) – Option value.

Raises

- KeyError – An invalid option name was supplied.
- TypeError – Option value was not the correct type.

msticnb.options.**show**()
Show help for options.

4.1.10 msticnb.read_modules module

read_modules - handles reading notebooklets modules.

class msticnb.read_modules.**FindResult**(*full_match*, *match_count*, *name*, *nb_class*)
Bases: tuple

Create new instance of FindResult(full_match, match_count, name, nb_class)

count()
Return number of occurrences of value.

full_match
Alias for field number 0

index()
Return first index of value.

Raises ValueError if the value is not present.

match_count
Alias for field number 1

name
Alias for field number 2

nb_class
Alias for field number 3

msticnb.read_modules.**discover_modules**(*nb_path: Union[str, Iterable[str]] = None*) → msticnb.common.NBContainer

Discover notebooks modules.

Parameters **nb_path** (*Union[str, Iterable[str]], optional*) – Additional path to search for notebooklets, by default None

Returns Container of notebooklets. This is structured as a tree mirroring the source folder names.

Return type *NBContainer*

msticnb.read_modules.**find**(*keywords: str, full_match=True*) → List[Tuple[str, msticnb.notebooklet.Notebooklet]]

Search for Notebooklets matching key words.

Parameters

- **keywords** (*str*) – Space or comma-separated words to search for. Terms can be regular expressions.
- **full_match** (*bool*) – If True only return full matches, default is True. If False it will return partial matches.

Returns List of matches sorted by closest match

Return type List[Tuple[str, *Notebooklet*]]

Notes

Search terms are treated as regular expressions, so any regular expression reserved characters will be treated as part of the regex pattern.

4.1.11 msticnb.dataviewers module

Data viewers mixin classes.

class msticnb.data_viewers.**DFViewer**
Bases: object

Mixin class for NotebookletResult.

view_events(*summary_cols: List[str] = None, attrib: Optional[str] = None, data: Optional[pandas.core.frame.DataFrame] = None, **kwargs*) → msticpy.nbtools.nbwidgets.select_item.SelectItem

Return simple data view for DataFrame/result attribute.

Parameters

- **summary_cols** (*List[str], optional*) – [description]
- **attrib** (*Optional[str], optional*) – [description], by default None
- **data** (*Optional[pd.DataFrame], optional*) – [description], by default None

- **kwargs** – Additional keyword arguments passed to the SelectItem widget.

Returns Browser for events in DataFrame.

Return type SelectItem

Raises

- AttributeError – Attribute name not in results class.
- TypeError – Input data or attribute is not a DataFrame
- MsticnbMissingParameterError – One of *data* or *attrib* parameters must be supplied
- KeyError – Summary column name specified that isn't in the DataFrame

4.2 Notebooklets source documentation

4.2.1 Categories

Azure Sentinel Notebooklets

Subpackages

Account notebooklets

msticnb.nb.azsent.account.account_summary	Notebooklet for Account Summary.

Submodules

msticnb.nb.azsent.account.account_summary module

Notebooklet for Account Summary.

class msticnb.nb.azsent.account.account_summary.**AccountSummary**(**args*, ***kwargs*)

Bases: *msticnb.notebooklet.Notebooklet*

Retrieves account summary for the selected account.

Main operations: - Searches for matches for the account name in Active Directory,

Windows and Linux host logs.

- If one or more matches are found it will return a selection widget that you can use to pick the account.
- Selecting the account displays a summary of recent activity and retrieves any alerts and hunting bookmarks related to the account
- The alerts and bookmarks are browseable using the *browse_alerts* and *browse_bookmarks* methods
- You can call the *get_additional_data* method to retrieve and display more detailed activity information for the account.

All of the returned data items are stored in the results class as entities, pandas DataFrames or Bokeh visualizations. Run help(nblt) on the notebooklet class to see usage. Run help(result) on the result class to see documentation of its properties. Run the print_options() method on either the notebooklet or results class to see information about the *options* parameter for the run() method.

- get_alerts: Retrieve alerts and display timeline for the account.
- get_bookmarks: Retrieve investigation bookmarks for the account

None

Initialize the Account Summary notebooklet.

ACCOUNT_TYPE
alias of *AccountType*

classmethod all_options() → List[str]
Return supported options for Notebooklet run function.

Returns Supported options.

Return type List[str]

az_activity_timeline_by_ip()
Display Azure activity timeline by IP address.

az_activity_timeline_by_operation()
Display Azure activity timeline by operation.

az_activity_timeline_by_provider()
Display Azure activity timeline by provider.

browse_accounts() → msticpy.nbtools.nbwidgets.select_item.SelectItem
Return the accounts browser/viewer.

browse_alerts() → msticpy.nbtools.nbwidgets.select_alert.SelectAlert
Return alert browser/viewer.

browse_bookmarks() → msticpy.nbtools.nbwidgets.select_item.SelectItem
Return bookmark browser/viewer.

check_table_exists(*table: str*) → bool
Check to see if the table exists in the provider.

Parameters **table** (*str*) – Table name

Returns True if the table exists, otherwise False.

Return type bool

check_valid_result_data(*attrib: str = None*, *silent: bool = False*) → bool
Check that the result is valid and *attrib* contains data.

Parameters

- **attrib** (*str*) – Name of the attribute to check, if None this function only checks for a valid _last_result.
- **silent** (*bool*) – If True, suppress output.

Returns Returns True if valid data is available, else False.

Return type bool

classmethod default_options() → List[str]
Return default options for Notebooklet run function.

Returns Supported options.

Return type List[str]

classmethod description() → str
Return description of the Notebooklet.

Returns Description

Return type str

display_alert_timeline()
Display the alert timeline.

classmethod entity_types() → List[str]
Entity types supported by the notebooklet.

Returns Entity names

Return type List[str]

get_additional_data() → pandas.core.frame.DataFrame
Find additional data for the selected account.

Returns Results with expanded columns.

Return type pd.DataFrame

get_geoip_map()
Return Folium map of IP activity.

classmethod get_help(*fmt='html'*) → str
Return HTML document for class.

get_methods() → Dict[str, Callable[[Any], Any]]
Return methods available for this class.

get_pivot_run(*get_timespan: Callable[[], msticpy.common.timespan.TimeSpan]*)
Return Pivot-wrappable run function.

get_provider(*provider_name: str*)
Return data provider for the specified name.

Parameters **provider_name** (*str*) – Name of the provider

Returns Provider instance.

Return type Any

Raises MsticnbDataProviderError – If provider is not found.

classmethod get_settings(*print_settings=True*) → Optional[str]
Print or return metadata for class.

Parameters **print_settings** (*bool, optional*) – Print to standard, by default True or return the str formatted content.

Returns If *print_settings* is True, returns None. If False, returns LF-delimited string of metadata settings.

Return type Optional[str]

Notes

Use *metadata* attribute to retrieve the metadata directly.

host_logon_timeline()
Display IP address summary.

classmethod import_cell()
Import the text of this module into a new cell.

classmethod keywords() → List[str]
Return search keywords for Notebooklet.

Returns Keywords

Return type List[str]

list_methods() → List[str]
Return list of methods with descriptions.

classmethod list_options() → str
Return options document for Notebooklet run function.

Returns Supported options.

Return type List[str]

classmethod match_terms(*search_terms: str*) → Tuple[bool, int]
Search class definition for *search_terms*.

Parameters **search_terms** (*str*) – One or more search terms, separated by spaces or commas. Terms can be simple strings or regular expressions.

Returns Returns a tuple of bool (True if all terms match) and int (count of matched terms)

Return type Tuple[bool, int]

metadata = NBMetadata(name='AlertSummary', mod_name='msticnb.nb.azsent.account.account_

module_path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/msticnb/chec

classmethod name() → str
Return name of the Notebooklet.

Returns Name

Return type str

classmethod print_options()
Print options for Notebooklet run function.

result
Return result of the most recent notebooklet run.

Returns Notebooklet result class or None if nothing has been run.

Return type Optional[*NotebookletResult*]

run(*value: Any = None*, *data: Optional[pandas.core.frame.DataFrame] = None*, *timespan: Optional[msticpy.common.timespan.TimeSpan] = None*, *options: Optional[Iterable[str]] = None*, ***kwargs*) → msticnb.nb.azsent.account.account_summary.AccountSummaryResult
Return account activity summary.

Parameters

- **value** (*str*) – Account name to search for.

- **data** (*Optional[pd.DataFrame], optional*) – Not used.
- **timespan** (*TimeSpan*) – Timespan for queries
- **options** (*Optional[Iterable[str]], optional*) – List of options to use, by default None. A value of None means use default options. Options prefixed with "+" will be added to the default options. To see the list of available options type *help(cls)* where "cls" is the notebooklet class or an instance of this class.
- **account_types** (*Iterable[AccountType], Optional*) – A list of account types to search for, by default all types.

Returns Result object with attributes for each result type.

Return type *AccountSummaryResult*

Raises MsticnbMissingParameterError – If required parameters are missing

classmethod show_help()
Display Documentation for class.

show_ip_summary()
Display Azure activity timeline by operation.

silent
Get the current instance setting for silent running.

Returns Silent running is enabled.

Return type Optional[bool]

class msticnb.nb.azsent.account.account_summary.**AccountSummaryResult**(*description: Optional[str] = None, timespan: Optional[msticpy.common.timespan.TimeSpan] = None, notebooklet: Optional[Notebooklet] = None*)

Bases: *msticnb.notebooklet_result.NotebookletResult*

Account Summary Result.

account_activity
DataFrame of most recent activity.

Type pd.DataFrame

account_selector
Selection widget for accounts.

Type msticpy.nbtools.nbwidgets.SelectString

related_alerts
Alerts related to the account.

Type pd.DataFrame

alert_timeline
Timeline of alerts.

Type LayoutDOM

related_bookmarks
Investigation bookmarks related to the account.

Type pd.DataFrame

host_logons
Host logon attemtps for selected account.

Type pd.DataFrame

host_logon_summary
Host logon summary for selected account.

Type pd.DataFrame

azure_activity
Azure Account activity for selected account.

Type pd.DataFrame

account_activity_summary
Azure activity summary.

Type pd.DataFrame

azure_timeline_by_provider
Azure activity timeline grouped by provider

Type LayoutDOM

account_timeline_by_ip
Host or Azure activity timeline by IP Address.

Type LayoutDOM

azure_timeline_by_operation
Azure activity timeline grouped by operation

Type LayoutDOM

ip_address_summary
Summary of IP address properties and usage for the current activity.

Type pd.DataFrame

ip_all_data
Full details of operations with IP WhoIs and GeoIP data.

Type pd.DataFrame

Create new Notebooklet result instance.

Parameters

- **description** (*Optional[str], optional*) – Result description, by default None

- **timespan** (*Optional[TimeSpan], optional*) – TimeSpan for the results, by default None
- **notebooklet** (*Optional[, optional*) – Originating notebooklet, by default None

data_properties(*empty: bool = False*) → List[str]
Return list of attributes with populated data.

prop_doc(*name*) → Tuple[str, str]
Get the property documentation for the property.

properties
Return names of all properties.

view_events(*summary_cols: List[str] = None, attrib: Optional[str] = None, data: Optional[pandas.core.frame.DataFrame] = None, **kwargs*) → msticpy.nbtools.nbwidgets.select_item.SelectItem
Return simple data view for DataFrame/result attribute.

Parameters

- **summary_cols** (*List[str], optional*) – [description]
- **attrib** (*Optional[str], optional*) – [description], by default None
- **data** (*Optional[pd.DataFrame], optional*) – [description], by default None
- **kwargs** – Additional keyword arguments passed to the SelectItem widget.

Returns Browser for events in DataFrame.

Return type SelectItem

Raises

- `AttributeError` – Attribute name not in results class.
- `TypeError` – Input data or attribute is not a DataFrame
- `MsticnbMissingParameterError` – One of *data* or *attrib* parameters must be supplied
- `KeyError` – Summary column name specified that isn't in the DataFrame

vis_properties() → List[str]
Return list of properties with visualizations.

class msticnb.nb.azsent.account.account_summary.**AccountType**
Bases: `enum.Flag`

Account types.

All = 31

Azure = 7

AzureActiveDirectory = 1

AzureActivity = 2

Linux = 16

Office365 = 4

Windows = 8

in_list(*acct_types: Iterable[Union[AccountType, str]]*)
Is the current value in the *acct_types* list.

parse = <bound method AccountType.parse of <enum 'AccountType'>>

Alerts notebooklets

msticnb.nb.azsent.alert.ti_enrich	Alert TI enrichment - provides enrichment of alerts with threat intelligence.

Submodules

msticnb.nb.azsent.alert.ti_enrich module

Alert TI enrichment - provides enrichment of alerts with threat intelligence.

class msticnb.nb.azsent.alert.ti_enrich.**EnrichAlerts**(*data_providers: Optional[msticnb.data_providers.DataProviders] = None*, ***kwargs*)

Bases: *msticnb.notebooklet.Notebooklet*

Alert Enrichment Notebooklet Class.

Enriches Azure Sentinel alerts with TI data.

Intialize a new instance of the notebooklet class.

Parameters **data_providers** (*DataProviders, Optional*) – Optional DataProviders instance to query data. Most classes require this.

Raises MsticnbDataProviderError – If DataProviders has not been initialized. If required providers are specified by the notebooklet but are not available.

classmethod all_options() → List[str]
Return supported options for Notebooklet run function.

Returns Supported options.

Return type List[str]

check_table_exists(*table: str*) → bool
Check to see if the table exists in the provider.

Parameters **table** (*str*) – Table name

Returns True if the table exists, otherwise False.

Return type bool

check_valid_result_data(*attrib: str = None*, *silent: bool = False*) → bool
Check that the result is valid and *attrib* contains data.

Parameters

- **attrib** (*str*) – Name of the attribute to check, if None this function only checks for a valid _last_result.
- **silent** (*bool*) – If True, suppress output.

Returns Returns True if valid data is available, else False.

Return type bool

classmethod default_options() → List[str]
Return default options for Notebooklet run function.

Returns Supported options.

Return type List[str]

classmethod description() → str
Return description of the Notebooklet.

Returns Description

Return type str

classmethod entity_types() → List[str]
Entity types supported by the notebooklet.

Returns Entity names

Return type List[str]

classmethod get_help(*fmt='html'*) → str
Return HTML document for class.

get_methods() → Dict[str, Callable[[Any], Any]]
Return methods available for this class.

get_pivot_run(*get_timespan: Callable[[], msticpy.common.timespan.TimeSpan]*)
Return Pivot-wrappable run function.

get_provider(*provider_name: str*)
Return data provider for the specified name.

Parameters **provider_name** (*str*) – Name of the provider

Returns Provider instance.

Return type Any

Raises MsticnbDataProviderError – If provider is not found.

classmethod get_settings(*print_settings=True*) → Optional[str]
Print or return metadata for class.

Parameters **print_settings** (*bool, optional*) – Print to standard, by default True or return the str formatted content.

Returns If *print_settings* is True, returns None. If False, returns LF-delimited string of metadata settings.

Return type Optional[str]

Notes

Use *metadata* attribute to retrieve the metadata directly.

classmethod import_cell()
Import the text of this module into a new cell.

classmethod keywords() → List[str]
Return search keywords for Notebooklet.

Returns Keywords

Return type List[str]

list_methods() → List[str]
Return list of methods with descriptions.

classmethod list_options() → str
Return options document for Notebooklet run function.

Returns Supported options.

Return type List[str]

classmethod match_terms(*search_terms: str*) → Tuple[bool, int]
Search class definition for *search_terms*.

Parameters **search_terms** (*str*) – One or more search terms, separated by spaces or commas. Terms can be simple strings or regular expressions.

Returns Returns a tuple of bool (True if all terms match) and int (count of matched terms)

Return type Tuple[bool, int]

metadata = NBMetadata(name='EnrichAlerts', mod_name='msticnb.nb.azsent.alert.ti_enrich

module_path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/msticnb/chec

classmethod name() → str
Return name of the Notebooklet.

Returns Name

Return type str

classmethod print_options()
Print options for Notebooklet run function.

result
Return result of the most recent notebooklet run.

Returns Notebooklet result class or None if nothing has been run.

Return type Optional[*NotebookletResult*]

run(*value: Optional[str] = None*, *data: Optional[pandas.core.frame.DataFrame] = None*, *timespan: Optional[msticpy.common.timespan.TimeSpan] = None*, *options: Optional[Iterable[str]] = None*, ***kwargs*) → msticnb.nb.azsent.alert.ti_enrich.TIEnrichResult
Return an enriched set of Alerts.

Parameters

- **timespan** (*TimeSpan*) – Timespan for queries
- **options** (*Optional[Iterable[str]], optional*) – List of options to use, by default None. A value of None means use default options. Options prefixed with "+" will be added to the default options. To see the list of available options type *help(cls)* where "cls" is the notebooklet class or an instance of this class.
- **value** (*Optional[str], optional*) – If you want to filter Alerts based on a specific entity specify it as a string.
- **data** (*Optional[pd.DataFrame], optional*) – If you have alerts in a DataFrame you can pass them rather than having the notebooklet query alerts.

Returns Result object with attributes for each result type.

Return type *TIEnrichResult*

Raises

- `MsticnbMissingParameterError` – If required parameters are missing
- `MsticnbDataProviderError` – If data is not avaliable

classmethod show_help()
Display Documentation for class.

silent
Get the current instance setting for silent running.

Returns Silent running is enabled.

Return type Optional[bool]

class msticnb.nb.azsent.alert.ti_enrich.**TIEnrichResult**(*description: Optional[str] = None, timespan: Optional[msticpy.common.timespan.TimeSpan] = None, notebooklet: Optional[Notebooklet] = None*)
Bases: *msticnb.notebooklet_result.NotebookletResult*

Template Results.

enriched_results
Alerts with additional TI enrichment

Type pd.DataFrame

picker
Alert picker

Type SelectAlert

Create new Notebooklet result instance.

Parameters

- **description** (*Optional[str], optional*) – Result description, by default None
- **timespan** (*Optional[TimeSpan], optional*) – TimeSpan for the results, by default None
- **notebooklet** (*Optional[, optional*) – Originating notebooklet, by default None

data_properties(*empty: bool = False*) → List[str]
Return list of attributes with populated data.

prop_doc(*name*) → Tuple[str, str]
Get the property documentation for the property.

properties
Return names of all properties.

view_events(*summary_cols: List[str] = None, attrib: Optional[str] = None, data: Optional[pandas.core.frame.DataFrame] = None, **kwargs*) → msticpy.nbtools.nbwidgets.select_item.SelectItem
Return simple data view for DataFrame/result attribute.

Parameters

- **summary_cols** (*List[str], optional*) – [description]
- **attrib** (*Optional[str], optional*) – [description], by default None
- **data** (*Optional[pd.DataFrame], optional*) – [description], by default None

- **kwargs** – Additional keyword arguments passed to the SelectItem widget.

Returns Browser for events in DataFrame.

Return type SelectItem

Raises

- AttributeError – Attribute name not in results class.
- TypeError – Input data or attribute is not a DataFrame
- MsticnbMissingParameterError – One of *data* or *attrib* parameters must be supplied
- KeyError – Summary column name specified that isn't in the DataFrame

vis_properties() → List[str]
Return list of properties with visualizations.

Host notebooklets

msticnb.nb.azsent.host.host_logons_summary	logons_summary - provides overview of host logon events.
msticnb.nb.azsent.host.host_network_summary	Notebooklet for Host Summary.
msticnb.nb.azsent.host.host_summary	Notebooklet for Host Summary.
msticnb.nb.azsent.host.win_host_events	Notebooklet for Windows Security Events.

Submodules

msticnb.nb.azsent.host.host_logons_summary module

logons_summary - provides overview of host logon events.

class msticnb.nb.azsent.host.host_logons_summary.**HostLogonsSummary**(*data_providers: Optional[msticnb.data_providers.Data = None*, ***kwargs*)

Bases: *msticnb.notebooklet.Notebooklet*

Host Logons Summary Notebooket class.

Queries and displays information about logons to a host including:

- Summary of sucessfull logons
- Visualizations of logon event times
- Geolocation of remote logon sources
- Visualizations of various logon elements depending on host type
- Data on users with failed and sucessful logons

Intialize a new instance of the notebooklet class.

Parameters **data_providers** (*DataProviders, Optional*) – Optional DataProviders instance to query data. Most classes require this.

Raises MsticnbDataProviderError – If DataProviders has not been initialized. If required providers are specified by the notebooklet but are not available.

classmethod all_options() → List[str]
Return supported options for Notebooklet run function.

Returns Supported options.

Return type List[str]

check_table_exists(*table: str*) → bool
Check to see if the table exists in the provider.

Parameters **table** (*str*) – Table name

Returns True if the table exists, otherwise False.

Return type bool

check_valid_result_data(*attrib: str = None*, *silent: bool = False*) → bool
Check that the result is valid and *attrib* contains data.

Parameters

- **attrib** (*str*) – Name of the attribute to check, if None this function only checks for a valid _last_result.
- **silent** (*bool*) – If True, suppress output.

Returns Returns True if valid data is available, else False.

Return type bool

classmethod default_options() → List[str]
Return default options for Notebooklet run function.

Returns Supported options.

Return type List[str]

classmethod description() → str
Return description of the Notebooklet.

Returns Description

Return type str

classmethod entity_types() → List[str]
Entity types supported by the notebooklet.

Returns Entity names

Return type List[str]

classmethod get_help(*fmt='html'*) → str
Return HTML document for class.

get_methods() → Dict[str, Callable[[Any], Any]]
Return methods available for this class.

get_pivot_run(*get_timespan: Callable[[], msticpy.common.timespan.TimeSpan]*)
Return Pivot-wrappable run function.

get_provider(*provider_name: str*)
Return data provider for the specified name.

Parameters **provider_name** (*str*) – Name of the provider

Returns Provider instance.

Return type Any

Raises MsticnbDataProviderError – If provider is not found.

classmethod get_settings(*print_settings=True*) → Optional[str]
Print or return metadata for class.

Parameters **print_settings** (*bool, optional*) – Print to standard, by default True or return the str formatted content.

Returns If *print_settings* is True, returns None. If False, returns LF-delimited string of metadata settings.

Return type Optional[str]

Notes

Use *metadata* attribute to retrieve the metadata directly.

classmethod import_cell()
Import the text of this module into a new cell.

classmethod keywords() → List[str]
Return search keywords for Notebooklet.

Returns Keywords

Return type List[str]

list_methods() → List[str]
Return list of methods with descriptions.

classmethod list_options() → str
Return options document for Notebooklet run function.

Returns Supported options.

Return type List[str]

classmethod match_terms(*search_terms: str*) → Tuple[bool, int]
Search class definition for *search_terms*.

Parameters **search_terms** (*str*) – One or more search terms, separated by spaces or commas. Terms can be simple strings or regular expressions.

Returns Returns a tuple of bool (True if all terms match) and int (count of matched terms)

Return type Tuple[bool, int]

metadata = NBMetadata(name='HostLogonsSummary', mod_name='msticnb.nb.azsent.host.host_

module_path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/msticnb/chec

classmethod name() → str
Return name of the Notebooklet.

Returns Name

Return type str

classmethod print_options()
Print options for Notebooklet run function.

result
Return result of the most recent notebooklet run.

Returns Notebooklet result class or None if nothing has been run.

Return type Optional[*NotebookletResult*]

run(*value: Any = None*, *data: Optional[pandas.core.frame.DataFrame] = None*, *timespan: Optional[msticpy.common.timespan.TimeSpan] = None*, *options: Optional[Iterable[str]] = None*, ***kwargs*) → msticnb.nb.azsent.host.host_logons_summary.HostLogonsSummaryResult
Return host summary data.

Parameters

- **value** (*str*) – Host name
- **data** (*Optional[pd.DataFrame], optional*) – Optionally pass raw data to use for analysis, by default None
- **timespan** (*TimeSpan*) – Timespan over which operations such as queries will be performed, by default None. This can be a TimeStamp object or another object that has valid *start*, *end*, or *period* attributes. Alternatively you can pass *start* and *end* datetime objects.
- **options** (*Optional[Iterable[str]], optional*) – List of options to use, by default None A value of None means use default options.

Returns Result object with attributes for each result type.

Return type HostLogonsSummaryResults

Raises

- MsticnbMissingParameterError – If required parameters are missing
- MsticnbDataProviderError – If data is not avaliable

classmethod show_help()
Display Documentation for class.

silent
Get the current instance setting for silent running.

Returns Silent running is enabled.

Return type Optional[bool]

class msticnb.nb.azsent.host.host_logons_summary.**HostLogonsSummaryResult**(*description: Optional[str] = None*, *timespan: Optional[msticpy.common.ti = None*, *notebooklet: Optional[Notebooklet] = None*)

Bases: *msticnb.notebooklet_result.NotebookletResult*

Host Logons Summary Results.

logon_sessions
A Dataframe summarizing all sucessfull and failed logon attempts observed during the specified time period.

Type pd.DataFrame

logon_map
A map showing remote logon attempt source locations. Red points represent failed logons, green successful.

Type FoliumMap

plots
A collection of Bokeh plot figures showing various aspects of observed logons. Keys are a descriptive name of the plot and values are the plot figures.

Type Dict

Create new Notebooklet result instance.

Parameters

- **description** (*Optional[str], optional*) – Result description, by default None
- **timespan** (*Optional[TimeSpan], optional*) – TimeSpan for the results, by default None
- **notebooklet** (*Optional[, optional*) – Originating notebooklet, by default None

data_properties(*empty: bool = False*) → List[str]
Return list of attributes with populated data.

prop_doc(*name*) → Tuple[str, str]
Get the property documentation for the property.

properties
Return names of all properties.

view_events(*summary_cols: List[str] = None*, *attrib: Optional[str] = None*, *data: Optional[pandas.core.frame.DataFrame] = None*, ***kwargs*) → msticpy.nbtools.nbwidgets.select_item.SelectItem

Return simple data view for DataFrame/result attribute.

Parameters

- **summary_cols** (*List[str], optional*) – [description]
- **attrib** (*Optional[str], optional*) – [description], by default None
- **data** (*Optional[pd.DataFrame], optional*) – [description], by default None
- **kwargs** – Additional keyword arguments passed to the SelectItem widget.

Returns Browser for events in DataFrame.

Return type SelectItem

Raises

- `AttributeError` – Attribute name not in results class.
- `TypeError` – Input data or attribute is not a DataFrame
- `MsticnbMissingParameterError` – One of *data* or *attrib* parameters must be supplied
- `KeyError` – Summary column name specified that isn't in the DataFrame

vis_properties() → List[str]

Return list of properties with visualizations.

msticnb.nb.azsent.host.host_summary module

Notebooklet for Host Summary.

class msticnb.nb.azsent.host.host_summary.**HostSummary**(*data_providers: Optional[msticnb.data_providers.DataProviders] = None*, ***kwargs*)

Bases: *msticnb.notebooklet.Notebooklet*

HostSummary Notebooklet class.

Queries and displays information about a host including:

- IP address assignment
- Related alerts
- Related hunting/investigation bookmarks
- Azure subscription/resource data.

- heartbeat: Query Heartbeat table for host information.
- azure_net: Query AzureNetworkAnalytics table for host network topology information.
- alerts: Query any alerts for the host.
- bookmarks: Query any bookmarks for the host.
- azure_api: Query Azure API for VM information.

None

Intialize a new instance of the notebooklet class.

Parameters **data_providers** (*DataProviders, Optional*) – Optional DataProviders instance to query data. Most classes require this.

Raises MsticnbDataProviderError – If DataProviders has not been initialized. If required providers are specified by the notebooklet but are not available.

classmethod all_options() → List[str]
Return supported options for Notebooklet run function.

Returns Supported options.

Return type List[str]

browse_alerts() → msticpy.nbtools.nbwidgets.select_alert.SelectAlert
Return alert browser/viewer.

check_table_exists(*table: str*) → bool
Check to see if the table exists in the provider.

Parameters **table** (*str*) – Table name

Returns True if the table exists, otherwise False.

Return type bool

check_valid_result_data(*attrib: str = None*, *silent: bool = False*) → bool
Check that the result is valid and *attrib* contains data.

Parameters

- **attrib** (*str*) – Name of the attribute to check, if None this function only checks for a valid _last_result.
- **silent** (*bool*) – If True, suppress output.

Returns Returns True if valid data is available, else False.

Return type bool

classmethod default_options() → List[str]
Return default options for Notebooklet run function.

Returns Supported options.

Return type List[str]

classmethod description() → str
Return description of the Notebooklet.

Returns Description

Return type str

display_alert_timeline()
Display the alert timeline.

classmethod entity_types() → List[str]
Entity types supported by the notebooklet.

Returns Entity names

Return type List[str]

classmethod get_help(*fmt='html'*) → str
Return HTML document for class.

get_methods() → Dict[str, Callable[[Any], Any]]
Return methods available for this class.

get_pivot_run(*get_timespan: Callable[[], msticpy.common.timespan.TimeSpan]*)
Return Pivot-wrappable run function.

get_provider(*provider_name: str*)
Return data provider for the specified name.

Parameters **provider_name** (*str*) – Name of the provider

Returns Provider instance.

Return type Any

Raises MsticnbDataProviderError – If provider is not found.

classmethod get_settings(*print_settings=True*) → Optional[str]
Print or return metadata for class.

Parameters **print_settings** (*bool, optional*) – Print to standard, by default True or return the str formatted content.

Returns If *print_settings* is True, returns None. If False, returns LF-delimited string of metadata settings.

Return type Optional[str]

Notes

Use *metadata* attribute to retrieve the metadata directly.

classmethod import_cell()
Import the text of this module into a new cell.

classmethod keywords() → List[str]
Return search keywords for Notebooklet.

Returns Keywords

Return type List[str]

list_methods() → List[str]
Return list of methods with descriptions.

classmethod list_options() → str
Return options document for Notebooklet run function.

Returns Supported options.

Return type List[str]

classmethod match_terms(*search_terms: str*) → Tuple[bool, int]
Search class definition for *search_terms*.

Parameters **search_terms** (*str*) – One or more search terms, separated by spaces or commas. Terms can be simple strings or regular expressions.

Returns Returns a tuple of bool (True if all terms match) and int (count of matched terms)

Return type Tuple[bool, int]

metadata = NBMetadata(name='HostSummary', mod_name='msticnb.nb.azsent.host.host_summary

module_path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/msticnb/chec

classmethod name() → str
Return name of the Notebooklet.

Returns Name

Return type str

classmethod print_options()
Print options for Notebooklet run function.

result
Return result of the most recent notebooklet run.

Returns Notebooklet result class or None if nothing has been run.

Return type Optional[*NotebookletResult*]

run(*value: Any = None*, *data: Optional[pandas.core.frame.DataFrame] = None*, *timespan: Optional[msticpy.common.timespan.TimeSpan] = None*, *options: Optional[Iterable[str]] = None*, ***kwargs*) → msticnb.nb.azsent.host.host_summary.HostSummaryResult
Return host summary data.

Parameters

- **value** (*str*) – Host name
- **data** (*Optional[pd.DataFrame], optional*) – Not used, by default None
- **timespan** (*TimeSpan*) – Timespan over which operations such as queries will be performed, by default None. This can be a TimeStamp object or another object that has valid *start*, *end*, or *period* attributes.
- **options** (*Optional[Iterable[str]], optional*) – List of options to use, by default None A value of None means use default options. Options prefixed with "+" will be added to the default options. To see the list of available options type *help(cls)* where "cls" is the notebooklet class or an instance of this class.

Other Parameters

- **start** (*Union[datetime, datelike-string]*) – Alternative to specifying timespan parameter.
- **end** (*Union[datetime, datelike-string]*) – Alternative to specifying timespan parameter.

Returns Result object with attributes for each result type.

Return type *HostSummaryResult*

Raises MsticnbMissingParameterError – If required parameters are missing

classmethod show_help()
Display Documentation for class.

silent
Get the current instance setting for silent running.

Returns Silent running is enabled.

Return type Optional[bool]

class msticnb.nb.azsent.host.host_summary.**HostSummaryResult**(*description: Optional[str] = None, timespan: Optional[msticpy.common.timespan.TimeSpan] = None, notebooklet: Optional[Notebooklet] = None*)

Bases: *msticnb.notebooklet_result.NotebookletResult*

Host Details Results.

host_entity
The host entity object contains data about the host such as name, environment, operating system version, IP addresses and Azure VM details. Depending on the type of host, not all of this data may be populated.

Type msticpy.datamodel.entities.Host

related_alerts
Pandas DataFrame of any alerts recorded for the host within the query time span.

Type pd.DataFrame

alert_timeline
Bokeh time plot of alerts recorded for host.

related_bookmarks
Pandas DataFrame of any investigation bookmarks relating to the host.

Type pd.DataFrame

events
Pandas DataFrame of any high severity events from the selected host.

Type pd.DataFrame

Create new Notebooklet result instance.

Parameters

- **description** (*Optional[str], optional*) – Result description, by default None
- **timespan** (*Optional[TimeSpan], optional*) – TimeSpan for the results, by default None
- **notebooklet** (*Optional[], optional*) – Originating notebooklet, by default None

data_properties(*empty: bool = False*) → List[str]
Return list of attributes with populated data.

prop_doc(*name*) → Tuple[str, str]
Get the property documentation for the property.

properties
Return names of all properties.

view_events(*summary_cols: List[str] = None, attrib: Optional[str] = None, data: Optional[pandas.core.frame.DataFrame] = None, **kwargs*) → msticpy.nbtools.nbwidgets.select_item.SelectItem
Return simple data view for DataFrame/result attribute.

Parameters

- **summary_cols** (*List[str], optional*) – [description]

- **attrib** (*Optional[str], optional*) – [description], by default None
- **data** (*Optional[pd.DataFrame], optional*) – [description], by default None
- **kwargs** – Additional keyword arguments passed to the SelectItem widget.

Returns Browser for events in DataFrame.

Return type SelectItem

Raises

- `AttributeError` – Attribute name not in results class.
- `TypeError` – Input data or attribute is not a DataFrame
- `MsticnbMissingParameterError` – One of *data* or *attrib* parameters must be supplied
- `KeyError` – Summary column name specified that isn't in the DataFrame

vis_properties() → List[str]
Return list of properties with visualizations.

msticnb.nb.azsent.host.host_network_summary module

Notebooklet for Host Summary.

class msticnb.nb.azsent.host.host_network_summary.**HostNetworkSummary**(*data_providers: Optional[msticnb.data_providers.D = None*, ***kwargs*)

Bases: *msticnb.notebooklet.Notebooklet*

HostSummary Notebooklet class.

Queries and displays information about a host including:

- IP address assignment
- Related alerts
- Related hunting/investigation bookmarks
- Azure subscription/resource data.
- map: Display a map of remote IP addresses communicating with the host.
- ti: Enrich network flow data with Threat Inteligence.
- whois: Enrich network flow data with WhoIs information.

None

Intialize a new instance of the notebooklet class.

Parameters **data_providers** (*DataProviders, Optional*) – Optional DataProviders instance to query data. Most classes require this.

Raises `MsticnbDataProviderError` – If DataProviders has not been initialized. If required providers are specified by the notebooklet but are not available.

classmethod all_options() → List[str]
Return supported options for Notebooklet run function.

Returns Supported options.

Return type List[str]

check_table_exists(*table: str*) → bool
Check to see if the table exists in the provider.

Parameters **table** (*str*) – Table name

Returns True if the table exists, otherwise False.

Return type bool

check_valid_result_data(*attrib: str = None*, *silent: bool = False*) → bool
Check that the result is valid and *attrib* contains data.

Parameters

- **attrib** (*str*) – Name of the attribute to check, if None this function only checks for a valid _last_result.
- **silent** (*bool*) – If True, suppress output.

Returns Returns True if valid data is available, else False.

Return type bool

classmethod default_options() → List[str]
Return default options for Notebooklet run function.

Returns Supported options.

Return type List[str]

classmethod description() → str
Return description of the Notebooklet.

Returns Description

Return type str

classmethod entity_types() → List[str]
Entity types supported by the notebooklet.

Returns Entity names

Return type List[str]

classmethod get_help(*fmt='html'*) → str
Return HTML document for class.

get_methods() → Dict[str, Callable[[Any], Any]]
Return methods available for this class.

get_pivot_run(*get_timespan: Callable[[], msticpy.common.timespan.TimeSpan]*)
Return Pivot-wrappable run function.

get_provider(*provider_name: str*)
Return data provider for the specified name.

Parameters **provider_name** (*str*) – Name of the provider

Returns Provider instance.

Return type Any

Raises `MsticnbDataProviderError` – If provider is not found.

classmethod get_settings(*print_settings=True*) → Optional[str]
Print or return metadata for class.

Parameters **print_settings** (*bool, optional*) – Print to standard, by default True or return the str formatted content.

Returns If *print_settings* is True, returns None. If False, returns LF-delimited string of metadata settings.

Return type Optional[str]

Notes

Use *metadata* attribute to retrieve the metadata directly.

classmethod import_cell()
Import the text of this module into a new cell.

classmethod keywords() → List[str]
Return search keywords for Notebooklet.

Returns Keywords

Return type List[str]

list_methods() → List[str]
Return list of methods with descriptions.

classmethod list_options() → str
Return options document for Notebooklet run function.

Returns Supported options.

Return type List[str]

classmethod match_terms(*search_terms: str*) → Tuple[bool, int]
Search class definition for *search_terms*.

Parameters **search_terms** (*str*) – One or more search terms, separated by spaces or commas. Terms can be simple strings or regular expressions.

Returns Returns a tuple of bool (True if all terms match) and int (count of matched terms)

Return type Tuple[bool, int]

metadata = NBMetadata(name='HostNetworkSummary', mod_name='msticnb.nb.azsent.host.host_

module_path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/msticnb/checl

classmethod name() → str
Return name of the Notebooklet.

Returns Name

Return type str

classmethod print_options()
Print options for Notebooklet run function.

result
Return result of the most recent notebooklet run.

Returns Notebooklet result class or None if nothing has been run.

Return type Optional[*NotebookletResult*]

run(*value: Any = None*, *data: Optional[pandas.core.frame.DataFrame] = None*, *timespan: Optional[msticpy.common.timespan.TimeSpan] = None*, *options: Optional[Iterable[str]] = None*, ***kwargs*) → msticnb.nb.azsent.host.host_network_summary.HostNetworkSummaryResult
Return host summary data.

Parameters

- **value** (*str*) – Host entity
- **data** (*Optional[pd.DataFrame], optional*) – Not used, by default None
- **timespan** (*TimeSpan*) – Timespan over which operations such as queries will be performed, by default None. This can be a TimeStamp object or another object that has valid *start*, *end*, or *period* attributes.
- **options** (*Optional[Iterable[str]], optional*) – List of options to use, by default None A value of None means use default options. Options prefixed with "+" will be added to the default options. To see the list of available options type *help(cls)* where "cls" is the notebooklet class or an instance of this class.

Other Parameters

- **start** (*Union[datetime, datelike-string]*) – Alternative to specifying timespan parameter.
- **end** (*Union[datetime, datelike-string]*) – Alternative to specifying timespan parameter.

Returns Result object with attributes for each result type.

Return type *HostSummaryResult*

Raises MsticnbMissingParameterError – If required parameters are missing

classmethod show_help()
Display Documentation for class.

silent
Get the current instance setting for silent running.

Returns Silent running is enabled.

Return type Optional[bool]

class msticnb.nb.azsent.host.host_network_summary.**HostNetworkSummaryResult**(*description: Optional[str] = None*, *timespan: Optional[msticpy.common.timespan.TimeSpan] = None*, *notebooklet: Optional[Notebooklet] = None*)

Bases: *msticnb.notebooklet_result.NotebookletResult*

Host Network Summary Results.

Create new Notebooklet result instance.

data_properties(*empty: bool = False*) → List[str]
Return list of attributes with populated data.

prop_doc(*name*) → Tuple[str, str]
Get the property documentation for the property.

properties
Return names of all properties.

view_events(*summary_cols: List[str] = None*, *attrib: Optional[str] = None*, *data: Optional[pandas.core.frame.DataFrame] = None*, ***kwargs*) → msticpy.nbtools.nbwidgets.select_item.SelectItem
Return simple data view for DataFrame/result attribute.

Parameters

- **summary_cols** (*List[str], optional*) – [description]
- **attrib** (*Optional[str], optional*) – [description], by default None
- **data** (*Optional[pd.DataFrame], optional*) – [description], by default None
- **kwargs** – Additional keyword arguments passed to the SelectItem widget.

Returns Browser for events in DataFrame.

Return type SelectItem

Raises

- `AttributeError` – Attribute name not in results class.
- `TypeError` – Input data or attribute is not a DataFrame
- `MsticnbMissingParameterError` – One of *data* or *attrib* parameters must be supplied
- `KeyError` – Summary column name specified that isn't in the DataFrame

vis_properties() → List[str]
Return list of properties with visualizations.

msticnb.nb.azsent.host.win_host_events module

Notebooklet for Windows Security Events.

class msticnb.nb.azsent.host.win_host_events.**WinHostEvents**(*data_providers: Optional[msticnb.data_providers.DataProviders] = None*, ***kwargs*)

Bases: *msticnb.notebooklet.Notebooklet*

Windows host Security Events Notebooklet class.

Queries and displays Windows Security Events including:

- All security events summary
- Extracting and displaying account management events
- Account management event timeline
- Optionally parsing packed event data into DataFrame columns

Process (4688) and Account Logon (4624, 4625) are not included in the event types processed by this module.

- event_pivot: Display a summary of all event types.
- acct_events: Display a summary and timeline of account management events.
- expand_events: parses the XML EventData column into separate DataFrame columns. This can be very expensive with a large event set. We recommend using the expand_events() method to select a specific subset of events to process.

Intialize a new instance of the notebooklet class.

Parameters **data_providers** (*DataProviders, Optional*) – Optional DataProviders instance to query data. Most classes require this.

Raises MsticnbDataProviderError – If DataProviders has not been initialized. If required providers are specified by the notebooklet but are not available.

classmethod all_options() → List[str]
Return supported options for Notebooklet run function.

Returns Supported options.

Return type List[str]

check_table_exists(*table: str*) → bool
Check to see if the table exists in the provider.

Parameters **table** (*str*) – Table name

Returns True if the table exists, otherwise False.

Return type bool

check_valid_result_data(*attrib: str = None*, *silent: bool = False*) → bool
Check that the result is valid and *attrib* contains data.

Parameters

- **attrib** (*str*) – Name of the attribute to check, if None this function only checks for a valid _last_result.
- **silent** (*bool*) – If True, suppress output.

Returns Returns True if valid data is available, else False.

Return type bool

classmethod default_options() → List[str]
Return default options for Notebooklet run function.

Returns Supported options.

Return type List[str]

classmethod description() → str
Return description of the Notebooklet.

Returns Description

Return type str

classmethod entity_types() → List[str]
Entity types supported by the notebooklet.

Returns Entity names

Return type List[str]

expand_events(*event_ids: Union[int, Iterable[int], None] = None*) → pandas.core.frame.DataFrame
Expand *EventData* for *event_ids* into separate columns.

Parameters **event_ids** (*Optional[Union[int, Iterable[int]]], optional*) – Single or interable of event IDs (ints). If no event_ids are specified all events will be expanded.

Returns Results with expanded columns.

Return type pd.DataFrame

Notes

For a specific event ID you can expand the EventProperties values into their own columns using this function. You can do this for the whole data set but it will time-consuming and result in a lot of sparse columns in the output data frame.

classmethod get_help(*fmt='html'*) → str
Return HTML document for class.

get_methods() → Dict[str, Callable[[Any], Any]]
Return methods available for this class.

get_pivot_run(*get_timespan: Callable[[], msticpy.common.timespan.TimeSpan]*)
Return Pivot-wrappable run function.

get_provider(*provider_name: str*)
Return data provider for the specified name.

Parameters **provider_name** (*str*) – Name of the provider

Returns Provider instance.

Return type Any

Raises MsticnbDataProviderError – If provider is not found.

classmethod get_settings(*print_settings=True*) → Optional[str]
Print or return metadata for class.

Parameters **print_settings** (*bool, optional*) – Print to standard, by default True or return the str formatted content.

Returns If *print_settings* is True, returns None. If False, returns LF-delimited string of metadata settings.

Return type Optional[str]

Notes

Use *metadata* attribute to retrieve the metadata directly.

classmethod import_cell()
Import the text of this module into a new cell.

classmethod keywords() → List[str]
Return search keywords for Notebooklet.

Returns Keywords

Return type List[str]

list_methods() → List[str]
Return list of methods with descriptions.

classmethod list_options() → str
Return options document for Notebooklet run function.

Returns Supported options.

Return type List[str]

classmethod match_terms(*search_terms: str*) → Tuple[bool, int]
Search class definition for *search_terms*.

Parameters **search_terms** (*str*) – One or more search terms, separated by spaces or commas. Terms can be simple strings or regular expressions.

Returns Returns a tuple of bool (True if all terms match) and int (count of matched terms)

Return type Tuple[bool, int]

metadata = NBMetadata(name='WinHostEvents', mod_name='msticnb.nb.azsent.host.win_host_

module_path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/msticnb/chec

classmethod name() → str
Return name of the Notebooklet.

Returns Name

Return type str

classmethod print_options()
Print options for Notebooklet run function.

result
Return result of the most recent notebooklet run.

Returns Notebooklet result class or None if nothing has been run.

Return type Optional[*NotebookletResult*]

run(*value: Any = None*, *data: Optional[pandas.core.frame.DataFrame] = None*, *timespan: Optional[msticpy.common.timespan.TimeSpan] = None*, *options: Optional[Iterable[str]] = None*, ***kwargs*) → msticnb.nb.azsent.host.win_host_events.WinHostEventsResult
Return Windows Security Event summary.

Parameters

- **value** (*str*) – Host name
- **data** (*Optional[pd.DataFrame], optional*) – Not used, by default None
- **timespan** (*TimeSpan*) – Timespan over which operations such as queries will be performed, by default None. This can be a TimeStamp object or another object that has valid *start*, *end*, or *period* attributes.
- **options** (*Optional[Iterable[str]], optional*) – List of options to use, by default None. A value of None means use default options. Options prefixed with "+" will be added to the default options. To see the list of available options type *help(cls)* where "cls" is the notebooklet class or an instance of this class.

Other Parameters

- **start** (*Union[datetime, datelike-string]*) – Alternative to specifying timespan parameter.

- **end** (*Union[datetime, datelike-string]*) – Alternative to specifying timespan parameter.

Returns Result object with attributes for each result type.

Return type *HostSummaryResult*

Raises `MsticnbMissingParameterError` – If required parameters are missing

classmethod show_help()
Display Documentation for class.

silent
Get the current instance setting for silent running.

Returns Silent running is enabled.

Return type Optional[bool]

class msticnb.nb.azsent.host.win_host_events.**WinHostEventsResult**(*description: Optional[str] = None, timespan: Optional[msticpy.common.timespan.Time = None, notebooklet: Optional[Notebooklet] = None*)

Bases: *msticnb.notebooklet_result.NotebookletResult*

Windows Host Security Events Results.

all_events
DataFrame of all raw events retrieved.

Type pd.DataFrame

event_pivot
DataFrame that is a pivot table of event ID vs. Account

Type pd.DataFrame

account_events
DataFrame containing a subset of account management events such as account and group modification.

Type pd.DataFrame

acct_pivot
DataFrame that is a pivot table of event ID vs. Account of account management events

Type pd.DataFrame

account_timeline
Bokeh plot figure or Layout showing the account events on an interactive timeline.

Type Union[Figure, LayoutDOM]

expanded_events
If *expand_events* option is specified, this will contain the parsed/expanded EventData as individual columns.

Type pd.DataFrame

Create new Notebooklet result instance.

Parameters

- **description** (*Optional[str], optional*) – Result description, by default None
- **timespan** (*Optional[TimeSpan], optional*) – TimeSpan for the results, by default None
- **notebooklet** (*Optional[, optional*) – Originating notebooklet, by default None

data_properties (*empty: bool = False*) → List[str]
Return list of attributes with populated data.

prop_doc (*name*) → Tuple[str, str]
Get the property documentation for the property.

properties
Return names of all properties.

view_events (*summary_cols: List[str] = None, attrib: Optional[str] = None, data: Optional[pandas.core.frame.DataFrame] = None, **kwargs*) → msticpy.nbtools.nbwidgets.select_item.SelectItem
Return simple data view for DataFrame/result attribute.

Parameters

- **summary_cols** (*List[str], optional*) – [description]
- **attrib** (*Optional[str], optional*) – [description], by default None
- **data** (*Optional[pd.DataFrame], optional*) – [description], by default None
- **kwargs** – Additional keyword arguments passed to the SelectItem widget.

Returns Browser for events in DataFrame.

Return type SelectItem

Raises

- `AttributeError` – Attribute name not in results class.
- `TypeError` – Input data or attribute is not a DataFrame
- `MsticnbMissingParameterError` – One of *data* or *attrib* parameters must be supplied
- `KeyError` – Summary column name specified that isn't in the DataFrame

vis_properties () → List[str]
Return list of properties with visualizations.

Network notebooklets

msticnb.nb.azsent.network. network_flow_summary	
msticnb.nb.azsent.network.ip_summary	IP Address Summary notebooklet.

Submodules

msticnb.nb.azsent.network.network_flow_summary module

msticnb.nb.azsent.network.ip_summary module

IP Address Summary notebooklet.

class msticnb.nb.azsent.network.ip_summary.**IpAddressSummary**(*data_providers: Optional[msticnb.data_providers.DataProviders = None*, ***kwargs*)

Bases: *msticnb.notebooklet.Notebooklet*

IP Address Summary Notebooklet class.

Queries and displays summary information about an IP address, including:

- Basic IP address properties
- IpAddress entity (and Host entity, if a host could be associated)
- WhoIs and Geo-location
- Azure activity and network data (optional)
- Office activity summary (optional)
- Threat intelligence reports
- Related alerts and hunting bookmarks

- geoip: Get geo location information for IP address.
- alerts: Get any alerts listing the IP address.
- host_logons: Find any hosts with logons using this IP address as a source.
- related_accounts: Find any accounts using this IP address in AAD or host logs.
- device_info: Find any devices associated with this IP address.
- device_network: Find any devices communicating with this IP address.

- bookmarks: Get any hunting bookmarks listing the IP address.
- heartbeat: Get the latest heartbeat record for for this IP address.
- az_net_if: Get the latest Azure network analytics interface data for this IP address.
- vmcomputer: Get the latest VMComputer record for this IP address.
- az_netflow: Get netflow information from AzureNetworkAnalytics table.
- passive_dns: Force fetching passive DNS data from a TI Provider even if IP is internal.
- az_activity: AAD sign-ins and Azure Activity logs.
- office_365: Office 365 activity.
- common_security: Get records from common security log.
- ti: Force get threat intelligence reports even for internal public IPs.

Intialize a new instance of the notebooklet class.

Parameters **data_providers** (*DataProviders, Optional*) – Optional DataProviders instance to query data. Most classes require this.

Raises MsticnbDataProviderError – If DataProviders has not been initialized. If required providers are specified by the notebooklet but are not available.

classmethod all_options() → List[str]
Return supported options for Notebooklet run function.

Returns Supported options.

Return type List[str]

browse_alerts() → msticpy.nbtools.nbwidgets.select_alert.SelectAlert
Return alert browser/viewer.

browse_ti_results()
Display Threat intel results.

check_table_exists(*table: str*) → bool
Check to see if the table exists in the provider.

Parameters **table** (*str*) – Table name

Returns True if the table exists, otherwise False.

Return type bool

check_valid_result_data(*attrib: str = None*, *silent: bool = False*) → bool
Check that the result is valid and *attrib* contains data.

Parameters

- **attrib** (*str*) – Name of the attribute to check, if None this function only checks for a valid _last_result.
- **silent** (*bool*) – If True, suppress output.

Returns Returns True if valid data is available, else False.

Return type bool

classmethod default_options() → List[str]
Return default options for Notebooklet run function.

Returns Supported options.

Return type List[str]

classmethod description() → str
Return description of the Notebooklet.

Returns Description

Return type str

display_alert_timeline()
Display the alert timeline.

classmethod entity_types() → List[str]
Entity types supported by the notebooklet.

Returns Entity names

Return type List[str]

classmethod get_help(*fmt='html'*) → str
Return HTML document for class.

get_methods() → Dict[str, Callable[[Any], Any]]
Return methods available for this class.

get_pivot_run(*get_timespan: Callable[[], msticpy.common.timespan.TimeSpan]*)
Return Pivot-wrappable run function.

get_provider(*provider_name: str*)
Return data provider for the specified name.

Parameters **provider_name** (*str*) – Name of the provider

Returns Provider instance.

Return type Any

Raises MsticnbDataProviderError – If provider is not found.

classmethod get_settings(*print_settings=True*) → Optional[str]
Print or return metadata for class.

Parameters **print_settings** (*bool, optional*) – Print to standard, by default True or return the str formatted content.

Returns If *print_settings* is True, returns None. If False, returns LF-delimited string of metadata settings.

Return type Optional[str]

Notes

Use *metadata* attribute to retrieve the metadata directly.

classmethod import_cell()
Import the text of this module into a new cell.

classmethod keywords() → List[str]
Return search keywords for Notebooklet.

Returns Keywords

Return type List[str]

list_methods() → List[str]
Return list of methods with descriptions.

classmethod list_options() → str
Return options document for Notebooklet run function.

Returns Supported options.

Return type List[str]

classmethod match_terms(*search_terms: str*) → Tuple[bool, int]
Search class definition for *search_terms*.

Parameters **search_terms** (*str*) – One or more search terms, separated by spaces or commas. Terms can be simple strings or regular expressions.

Returns Returns a tuple of bool (True if all terms match) and int (count of matched terms)

Return type Tuple[bool, int]

metadata = NBMetadata(name='IpAddressSummary', mod_name='msticnb.nb.azsent.network.ip_

module_path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/msticnb/chec

classmethod name() → str
Return name of the Notebooklet.

Returns Name

Return type str

netflow_by_direction() → bokeh.plotting.figure.Figure
Display netflows grouped by direction.

netflow_by_protocol() → bokeh.plotting.figure.Figure
Display netflows grouped by protocol.

netflow_total_by_protocol() → bokeh.plotting.figure.Figure
Display netflows grouped by protocol.

classmethod print_options()
Print options for Notebooklet run function.

result
Return result of the most recent notebooklet run.

Returns Notebooklet result class or None if nothing has been run.

Return type Optional[*NotebookletResult*]

run(*value: Any = None*, *data: Optional[pandas.core.frame.DataFrame] = None*, *timespan: Optional[msticpy.common.timespan.TimeSpan] = None*, *options: Optional[Iterable[str]] = None*, ***kwargs*) → msticnb.nb.azsent.network.ip_summary.IpSummaryResult
Return IP Address activity summary.

Parameters

- **value** (*str*) – IP Address - The key for searches
- **data** (*Optional[pd.DataFrame], optional*) – Not supported for this notebooklet.
- **timespan** (*TimeSpan*) – Timespan for queries
- **options** (*Optional[Iterable[str]], optional*) – List of options to use, by default None. A value of None means use default options. Options prefixed with "+" will be added to the default options. To see the list of available options type *help(cls)* where "cls" is the notebooklet class or an instance of this class.

Returns Result object with attributes for each result type.

Return type *IpSummaryResult*

Raises MsticnbMissingParameterError – If required parameters are missing

classmethod show_help()
Display Documentation for class.

silent
Get the current instance setting for silent running.

Returns Silent running is enabled.

Return type Optional[bool]

class msticnb.nb.azsent.network.ip_summary.**IpSummaryResult**(*description: Optional[str] = None, timespan: Optional[msticpy.common.timespan.TimeSpan] = None, notebooklet: Optional[Notebooklet] = None*)

Bases: *msticnb.notebooklet_result.NotebookletResult*

IPSummary Results.

ip_str
The input IP address as a string.

Type str

ip_address
Ip Address Python object

Type Optional[Union[IPv4Address, IPv6Address]]

ip_entity
IpAddress entity

Type IpAddress

ip_origin
"External" or "Internal"

Type str

host_entities
Host entity or entities associated with IP Address

Type Host

ip_type
IP address type - "Public", "Private", etc.

Type str

geoip
Geo location information as a dictionary.

Type Optional[Dict[str, Any]]

location
Location entity context object.

Type Optional[GeoLocation]

whois
WhoIs information for IP Address

Type pd.DataFrame

whois_nets
List of networks definitions from WhoIs data

Type pd.DataFrame

heartbeat
Heartbeat record for IP Address or host

Type pd.DataFrame

az_network_if
Azure NSG analytics interface record, if available

Type pd.DataFrame

vmcomputer
VMComputer latest record

Type pd.DataFrame

az_network_flows
Azure NSG flows for IP, if available

Type pd.DataFrame

az_network_flows_timeline
Azure NSG flows timeline, if data is available

Type Figure

aad_signins
AAD signin activity

Type pd.DataFrame = None

azure_activity
Azure Activity log entries

Type pd.DataFrame = None

azure_activity_summary
Azure Activity (AAD and Az Activity) summarized view

Type pd.DataFrame = None

office_activity
Office 365 activity

Type pd.DataFrame = None

common_security
Common Security Log entries for source IP

Type pd.DataFrame

related_bookmarks
Bookmarks related to IP Address

Type pd.DataFrame

alert_timeline
Timeline plot of alerts

Type Figure

ti_results
Threat intel lookup results

Type pd.DataFrame

passive_dns
Passive DNS lookup results

Type pd.DataFrame

`self.`**host_logons**
Hosts with logons from this IP Address

Type pd.DataFrame

`self.`**`related_accounts`**
Accounts with activity related to this IP Address

Type pd.DataFrame

`self.`**`associated_hosts`**
Hosts using this IP Address

Type pd.DataFrame

`self.`**`device_info`**
Device info of hosts using this IP Address

Type pd.DataFrame

`self.`**`network_connections`**
Network connections to/from this IP on other devices

Type pd.DataFrame = None

Create new IPSummaryResult result instance.

Parameters

- **`description`** (*`Optional[str], optional`*) – Result description, by default None
- **`timespan`** (*`Optional[TimeSpan], optional`*) – TimeSpan for the results, by default None
- **`notebooklet`** (*`Optional[, optional`*) – Originating notebooklet, by default None

`data_properties`(*empty: bool = False*) → List[str]
Return list of attributes with populated data.

`prop_doc`(*name*) → Tuple[str, str]
Get the property documentation for the property.

`properties`
Return names of all properties.

`view_events`(*summary_cols: List[str] = None, attrib: Optional[str] = None, data: Optional[pandas.core.frame.DataFrame] = None, **kwargs*) → msticpy.nbtools.nbwidgets.select_item.SelectItem
Return simple data view for DataFrame/result attribute.

Parameters

- **`summary_cols`** (*`List[str], optional`*) – [description]
- **`attrib`** (*`Optional[str], optional`*) – [description], by default None
- **`data`** (*`Optional[pd.DataFrame], optional`*) – [description], by default None
- **`kwargs`** – Additional keyword arguments passed to the SelectItem widget.

Returns Browser for events in DataFrame.

Return type SelectItem

Raises

- `AttributeError` – Attribute name not in results class.
- `TypeError` – Input data or attribute is not a DataFrame

- `MsticnbMissingParameterError` – One of *data* or *attrib* parameters must be supplied
- `KeyError` – Summary column name specified that isn't in the DataFrame

vis_properties() → List[str]
Return list of properties with visualizations.

Template notebooklet

Submodules

msticnb.nb.template.nb_template module

Template notebooklet.

Notebooklet modules have three main sections:

- **Result class definition**: This defines the attributes and descriptions of the data that you want to return from the notebooklet.
- **Notebooklet class definition**: This is the entry point for running the notebooklet. At minimum it should be a class derived from Notebooklet that implements a *run* method and returns your result class.
- **Functions**: These do most of the work of the notebooklet and usually the code that is copied from or adapted from the original notebook.

Having the latter section is optional. You can choose to implement this functionality in instance methods of the notebooklet class.

However, there are advantages to keeping these as separate functions outside the class. It means that all the data used in the functions has to be passed around as parameters and return values. This can improve the clarity of the code and reduce errors due to some dependency on some mysterious global state.

If the user of your notebooklet wants to import the module's code into a notebook to read and possibly adapt it, having standalone functions will make it easier from them understand and work with the code.

class msticnb.nb.template.nb_template.**TemplateNB**(*data_providers: Optional[msticnb.data_providers.DataProviders] = None*, ***kwargs*)

Bases: *msticnb.notebooklet.Notebooklet*

Template Notebooklet class.

Detailed description of things this notebooklet does:

- Fetches all events from XYZ
- Plots interesting stuff
- Returns extended metadata about the thing

Document the options that the Notebooklet takes, if any, Use these control which parts of the notebooklet get run.

- all_events: Gets all events about blah
- plot_events: Display and summary and timeline of events.
- get_metadata: fetches additional metadata about the entity

Intialize a new instance of the notebooklet class.

Parameters **data_providers** (*DataProviders, Optional*) – Optional DataProviders instance to query data. Most classes require this.

Raises MsticnbDataProviderError – If DataProviders has not been initialized. If required providers are specified by the notebooklet but are not available.

classmethod all_options() → List[str]
Return supported options for Notebooklet run function.

Returns Supported options.

Return type List[str]

check_table_exists(*table: str*) → bool
Check to see if the table exists in the provider.

Parameters **table** (*str*) – Table name

Returns True if the table exists, otherwise False.

Return type bool

check_valid_result_data(*attrib: str = None*, *silent: bool = False*) → bool
Check that the result is valid and *attrib* contains data.

Parameters

- **attrib** (*str*) – Name of the attribute to check, if None this function only checks for a valid _last_result.
- **silent** (*bool*) – If True, suppress output.

Returns Returns True if valid data is available, else False.

Return type bool

classmethod default_options() → List[str]
Return default options for Notebooklet run function.

Returns Supported options.

Return type List[str]

classmethod description() → str
Return description of the Notebooklet.

Returns Description

Return type str

classmethod entity_types() → List[str]
Entity types supported by the notebooklet.

Returns Entity names

Return type List[str]

classmethod get_help(*fmt='html'*) → str
Return HTML document for class.

get_methods() → Dict[str, Callable[[Any], Any]]
Return methods available for this class.

get_pivot_run(*get_timespan: Callable[[], msticpy.common.timespan.TimeSpan]*)
Return Pivot-wrappable run function.

get_provider(*provider_name: str*)
Return data provider for the specified name.

Parameters **provider_name** (*str*) – Name of the provider

Returns Provider instance.

Return type Any

Raises MsticnbDataProviderError – If provider is not found.

classmethod get_settings(*print_settings=True*) → Optional[str]
Print or return metadata for class.

Parameters **print_settings** (*bool, optional*) – Print to standard, by default True or return the str formatted content.

Returns If *print_settings* is True, returns None. If False, returns LF-delimited string of metadata settings.

Return type Optional[str]

Notes

Use *metadata* attribute to retrieve the metadata directly.

classmethod import_cell()
Import the text of this module into a new cell.

classmethod keywords() → List[str]
Return search keywords for Notebooklet.

Returns Keywords

Return type List[str]

list_methods() → List[str]
Return list of methods with descriptions.

classmethod list_options() → str
Return options document for Notebooklet run function.

Returns Supported options.

Return type List[str]

classmethod match_terms(*search_terms: str*) → Tuple[bool, int]
Search class definition for *search_terms*.

Parameters **search_terms** (*str*) – One or more search terms, separated by spaces or commas. Terms can be simple strings or regular expressions.

Returns Returns a tuple of bool (True if all terms match) and int (count of matched terms)

Return type Tuple[bool, int]

metadata = NBMetadata(name='TemplateNB', mod_name='msticnb.nb.template.nb_template', d

module_path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/msticnb/chec

classmethod name() → str
Return name of the Notebooklet.

Returns Name

Return type str

classmethod print_options()
Print options for Notebooklet run function.

result
Return result of the most recent notebooklet run.

Returns Notebooklet result class or None if nothing has been run.

Return type Optional[*NotebookletResult*]

run(*value: Any = None*, *data: Optional[pandas.core.frame.DataFrame] = None*, *timespan: Optional[msticpy.common.timespan.TimeSpan] = None*, *options: Optional[Iterable[str]] = None*, ***kwargs*) → msticnb.nb.template.nb_template.TemplateResult
Return XYZ summary.

Parameters

- **value** (*str*) – Host name - The key for searches - e.g. host, account, IPaddress
- **data** (*Optional[pd.DataFrame], optional*) – Alternatively use a DataFrame as input.
- **timespan** (*TimeSpan*) – Timespan for queries
- **options** (*Optional[Iterable[str]], optional*) – List of options to use, by default None. A value of None means use default options. Options prefixed with "+" will be added to the default options. To see the list of available options type *help(cls)* where "cls" is the notebooklet class or an instance of this class.

Returns Result object with attributes for each result type.

Return type *TemplateResult*

Raises MsticnbMissingParameterError – If required parameters are missing

run_additional_operation(*event_ids: Union[int, Iterable[int], None] = None*) → pandas.core.frame.DataFrame
Addition method.

Parameters **event_ids** (*Optional[Union[int, Iterable[int]]], optional*) – Single or interable of event IDs (ints).

Returns Results with expanded columns.

Return type pd.DataFrame

classmethod show_help()
Display Documentation for class.

silent
Get the current instance setting for silent running.

Returns Silent running is enabled.

Return type Optional[bool]

class msticnb.nb.template.nb_template.**TemplateResult**(*description: Optional[str] = None*, *timespan: Optional[msticpy.common.timespan.TimeSpan] = None*, *notebooklet: Optional[Notebooklet] = None*)
Bases: *msticnb.notebooklet_result.NotebookletResult*

Template Results.

all_events
DataFrame of all raw events retrieved.

Type pd.DataFrame

plot
Bokeh plot figure showing the account events on an interactive timeline.

Type bokeh.models.LayoutDOM

additional_info
Additional information for my notebooklet.

Type dict

Create new Notebooklet result instance.

Parameters

- **description** (*Optional[str], optional*) – Result description, by default None
- **timespan** (*Optional[TimeSpan], optional*) – TimeSpan for the results, by default None
- **notebooklet** (*Optional[, optional*) – Originating notebooklet, by default None

data_properties(*empty: bool = False*) → List[str]
Return list of attributes with populated data.

prop_doc(*name*) → Tuple[str, str]
Get the property documentation for the property.

properties
Return names of all properties.

view_events(*summary_cols: List[str] = None, attrib: Optional[str] = None, data: Optional[pandas.core.frame.DataFrame] = None, **kwargs*) → msticpy.nbtools.nbwidgets.select_item.SelectItem
Return simple data view for DataFrame/result attribute.

Parameters

- **summary_cols** (*List[str], optional*) – [description]
- **attrib** (*Optional[str], optional*) – [description], by default None
- **data** (*Optional[pd.DataFrame], optional*) – [description], by default None
- **kwargs** – Additional keyword arguments passed to the SelectItem widget.

Returns Browser for events in DataFrame.

Return type SelectItem

Raises

- `AttributeError` – Attribute name not in results class.
- `TypeError` – Input data or attribute is not a DataFrame
- `MsticnbMissingParameterError` – One of *data* or *attrib* parameters must be supplied
- `KeyError` – Summary column name specified that isn't in the DataFrame

vis_properties() → List[str]
Return list of properties with visualizations.

4.3 Notebook Common Library modules

4.3.1 Categories

Azure Sentinel library modules

Submodules

msticnb.nblib.azsent.host module

host_network_summary notebooklet.

class msticnb.nblib.azsent.host.**HostNameVerif**(*host_name*, *host_type*, *host_names*)
Bases: tuple

Create new instance of HostNameVerif(host_name, host_type, host_names)

count()
Return number of occurrences of value.

host_name
Alias for field number 0

host_names
Alias for field number 2

host_type
Alias for field number 1

index()
Return first index of value.

Raises ValueError if the value is not present.

msticnb.nblib.azsent.host.**get_aznet_topology**
Get Azure Network topology information for host or IP address.

Parameters

- **qry_prov** (*QueryProvider*) – Query provider to use for queries
- **host_entity** (*Host*) – Host entity to populate data with
- **host_name** (*str, optional*) – Host name, by default None
- **host_ip** (*str, optional*) – Host IP Address, by default None

msticnb.nblib.azsent.host.**get_heartbeat**
Get Heartbeat information for host or IP.

Parameters

- **qry_prov** (*QueryProvider*) – Query provider to use for queries
- **host_name** (*str, optional*) – Host name, by default None
- **host_ip** (*str, optional*) – Host IP Address, by default None

Returns Host entity

Return type Host

msticnb.nblib.azsent.host.**populate_host_entity**(*heartbeat_df: pandas.core.frame.DataFrame = None*, *az_net_df: pandas.core.frame.DataFrame = None*, *vmcomputer_df: pandas.core.frame.DataFrame = None*, *host_entity: msticpy.datamodel.entities.host.Host = None*, *geo_lookup: Any = None*) → msticpy.datamodel.entities.host.Host

Populate host with IP and other data.

Parameters

- **heartbeat_df** (*pd.DataFrame*) – Optional dataframe of heartbeat data for the host
- **az_net_df** (*pd.DataFrame*) – Optional dataframe of Azure network data for the host
- **vmcomputer_df** (*pd.DataFrame*) – Optional dataframe of VMComputer data for the host
- **host_entity** (*Host*) – Host entity in which to populate data. By default, a new host entity will be created.
- **geo_lookup** (*Any*) – GeoIP Provider to use, if needed.

Returns How with details of the IP data collected

Return type Host

msticnb.nblib.azsent.host.**verify_host_name**

Verify unique hostname by checking Win and Linux logs.

Parameters

- **qry_prov** (*QueryProvider*) – Kql query provider
- **timespan** (*TimeSpan*) – Time span over which to query
- **host_name** (*str*) – The full or partial hostname.

Returns Tuple[Optional[str], Optional[str], Optional[Dict[str, str]]] Named tuple HostNameVerif fields: host_name, host_type, host_names If unique hostname found, host_name is populated. If multiple matching hostnames found, host_names is populated and host_name is None. host_type is either Windows or Linux. If no matching host then all fields are None.

Return type *HostNameVerif*

msticnb.nblib.azsent.alert module

Alert utility functions.

msticnb.nblib.azsent.alert.**browse_alerts**(*nb_result*, *alert_attr='related_alerts'*) → msticpy.nbtools.nbwidgets.select_alert.SelectAlert

Return alert browser/viewer.

4.3.2 msticnb.nblib.iptools module

IP Helper functions.

msticnb.nblib.iptools.**arg_to_list**(*arg: Union[str, List[str]], delims=',; '*) → List[str]
Convert an optional list/str/str with delims into a list.

Parameters

- **arg** (*Union[str, List[str]]*) – A string, delimited string or list
- **delims** (*str, optional*) – The default delimiters to use, by default ",; "

Returns List of string components

Return type List[str]

Raises TypeError – If *arg* is not a string or list

msticnb.nblib.iptools.**convert_to_ip_entities**(*ip_str: Optional[str] = None, data: Optional[pandas.core.frame.DataFrame] = None, ip_col: Optional[str] = None, geo_lookup: Any = None*) → List[msticpy.datamodel.entities.ip_address.IpAddress]
Take in an IP Address string and converts it to an IP Entity.

Parameters

- **ip_str** (*str*) – A string with a single IP Address or multiple addresses delimited by comma or space
- **data** (*pd.DataFrame*) – Use DataFrame as input
- **ip_col** (*str*) – Column containing IP addresses
- **geo_lookup** (*bool*) – If true, do geolocation lookup on IPs, by default, True

Returns The populated IP entities including address and geo-location

Return type List

Raises ValueError – If neither ip_string or data/column provided as input

msticnb.nblib.iptools.**get_geoip_whois**(*geo_lookup, data: pandas.core.frame.DataFrame, ip_col: str*)
Get GeoIP and WhoIs data for IPs.

Parameters

- **geo_lookup** (*GeoIpLookup*) – GeoIP Provider
- **data** (*pd.DataFrame*) – Input data frame
- **ip_col** (*str*) – Name of Ip address column

Returns Results dataframe with GeoIP and WhoIs data

Return type pd.DataFrame

msticnb.nblib.iptools.**get_ip_ti**(*ti_lookup, data: pandas.core.frame.DataFrame, ip_col: str*) → pandas.core.frame.DataFrame
Lookup Threat Intel for IPAddress.

Parameters

- **ti_lookup** (*TILookup*) – TI Lookup provider
- **data** (*pd.DataFrame*) – Input data frame
- **ip_col** (*str*) – Name of Ip address column

Returns DataFrame with TI results for IPs

Return type pd.DataFrame

msticnb.nblib.iptools.**map_ips**(*data: pandas.core.frame.DataFrame*, *ip_col: str*, *summary_cols: Optional[List[str]] = None*, *geo_lookup: Any = None*) → msticpy.nbtools.foliummap.FoliumMap

Produce a map of IP locations.

Parameters

- **geo_lookup** (*Any*) – Geo-IP provider instance
- **data** (*pd.DataFrame*) – Data containing the IPAddress
- **ip_col** (*str*) – [description]
- **summary_cols** (*Optional[List[str]], optional*) – [description], by default None
- **geo_lookup** – GeoIP Provider instance.

Returns Folium map with items plotted.

Return type FoliumMap

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W