King Phisher Documentation

Release 1.8.0b0

Spencer McIntyre

May 23, 2017

Technical Documentation

1 The King Phisher Package 3

2 The King Phisher Client 99

3 The King Phisher Server 107

4 Plugins 117

5 Development References 123

6 Environment Variables 127

7 Change Log 129

8 Indices and tables 135

King Phisher REST API 137

Python Module Index 139

King Phisher is an open source Phishing Campaign Toolkit. This is its technical documentation intended for use by contributors. The source code is available on the GitHub homepage. Additionally documentation intended for use by users can be found in the King Phisher GitHub wiki.

CHAPTER 1

The King Phisher Package

client

This package contains all packages and modules specific to the client application.

client.assistants

assistants.campaign

Classes

class king_phisher.client.assistants.campaign.**CampaignAssistant**(*application*, *campaign_id=None*)

Bases: *king_phisher.client.gui_utilities.GladeGObject*

Display an assistant which walks the user through creating a new campaign or configuring an existing campaign. If no *campaign_id* is specified a new campaign will be created.

__init__(*application*, *campaign_id=None*)

Parameters

- **application** (*KingPhisherClientApplication*) – The application instance which this object belongs to.
- **campaign_id** – The ID of the campaign to edit.

campaign_name

The string value of the configured campaign name. This may be set even when the campaign was not created, which would be the case if the user closed the window.

client.dialogs

dialogs.about

Classes

class king_phisher.client.dialogs.about.**AboutDialog**(**args*, ***kwargs*)
Bases: *king_phisher.client.gui_utilities.GladeGObject*

Display a Gtk.AboutDialog with information regarding the King Phisher client.

dialogs.campaign_selection

Classes

class king_phisher.client.dialogs.campaign_selection.**CampaignSelectionDialog**(**args*, ***kwargs*)
Bases: *king_phisher.client.gui_utilities.GladeGObject*

Display a dialog which allows a new campaign to be created or an existing campaign to be opened.

load_campaigns()
Load campaigns from the remote server and populate the Gtk.TreeView.

dialogs.clone_page

Classes

class king_phisher.client.dialogs.clone_page.**ClonePageDialog**(**args*, ***kwargs*)
Bases: *king_phisher.client.gui_utilities.GladeGObject*

Display a dialog for cloning a web page. The logic for the cloning operation is provided by the *web_cloner* module.

dialogs.company_editor

Classes

class king_phisher.client.dialogs.company_editor.**CompanyEditorDialog**(**args*, ***kwargs*)
Bases: *king_phisher.client.gui_utilities.GladeGObject*

Display a dialog which can be used to edit the various fields associated with a company object.

dialogs.configuration

Classes

class king_phisher.client.dialogs.configuration.**ConfigurationDialog**(**args*, ***kwargs*)
Bases: *king_phisher.client.gui_utilities.GladeGObject*

Display the King Phisher client configuration dialog. Running this dialog via the `interact()` method will cause some server settings to be loaded.

dialogs.entry

Classes

class king_phisher.client.dialogs.entry.**TextEntryDialog**(**args*, ***kwargs*)

Bases: *king_phisher.client.gui_utilities.GladeGObject*

Display a `Gtk.Dialog` with a text entry suitable for prompting users for text input. If the user confirms the action, the text within the entry is returned. If the user cancels the action or closes the dialog, None is returned.

classmethod build_prompt(*application*, *title*, *label_text*, *entry_text=None*, *entry_tooltip_text=None*)

Create a *TextEntryDialog* instance configured with the specified text prompts.

Parameters

- **application** (`Gtk.Application`) – The parent application for this object.
- **title** (*str*) – The title to set for the dialog window.
- **label_text** (*str*) – The text to display in the entry's label.
- **entry_text** (*str*) – Text to place in the entry.
- **entry_tooltip_text** (*str*) – Text to display in the tool tip of the entry.

Returns If the prompt is submitted by the user, the text within the entry is returned.

Return type str

dialogs.exception

Functions

king_phisher.client.dialogs.exception.**format_exception_details**(*exc_type*, *exc_value*, *exc_traceback*, *error_uid=None*)

Format exception details to be show to a human. This should include enough information about the type of error that occurred and the system on which it was triggered to allow someone to attempt to debug and fix it. The first three parameters to this function directly correspond to the values returned from the `sys.exc_info()` function.

Parameters

- **exc_type** – The type of the exception.
- **exc_value** – The exception instance.
- **exc_traceback** – The traceback object corresponding to the exception.
- **error_uid** (str, `uuid.UUID`) – A unique identifier for this exception.

Returns A formatted message containing the details about the exception and environment.

Return type str

king_phisher.client.dialogs.exception.**format_exception_name**(*exc_type*)
Format the exception name into a more easily recognizable format.

Parameters **exc_type** – The type of the exception.

Returns The formatted exception name.

Return type str

Classes

class king_phisher.client.dialogs.exception.**ExceptionDialog**(*application*, *exc_info=None*, *error_uid=None*)
Bases: *king_phisher.client.gui_utilities.GladeGObject*

Display a dialog which shows an error message for a python exception. The dialog includes useful details for reporting and debugging the exception which occurred.

__init__(*application*, *exc_info=None*, *error_uid=None*)

Parameters

- **application** (Gtk.Application) – The parent application for this object.
- **exc_info** (*tuple*) – The exception information as provided by sys.exc_info().
- **error_uid** (*str*) – An optional unique identifier for the exception that can be provided for tracking purposes.

dialogs.login

Classes

class king_phisher.client.dialogs.login.**LoginDialogBase**(**args*, ***kwargs*)
Bases: *king_phisher.client.gui_utilities.GladeGObject*

This object is basic login dialog object that can be inherited from and customized.

class king_phisher.client.dialogs.login.**LoginDialog**(**args*, ***kwargs*)
Bases: *king_phisher.client.dialogs.login.LoginDialogBase*

This object is the main King Phisher login dialog, it is used to prompt for connection information for the King Phisher server.

It allows the user to specify the host and port to connect to and credentials for authentication.

class king_phisher.client.dialogs.login.**SMTPLoginDialog**(**args*, ***kwargs*)
Bases: *king_phisher.client.dialogs.login.LoginDialogBase*

This object is the King Phisher SMTP login dialog, it is used to prompt for connection information to an SMTP server.

It allows the user to specify the host and port to connect to and credentials for authentication.

class king_phisher.client.dialogs.login.**SSHLoginDialog**(**args*, ***kwargs*)
Bases: *king_phisher.client.dialogs.login.LoginDialogBase*

This object is the King Phisher SSH login dialog, it is used to prompt for connection information to an SSH server.

It allows the user to specify the host and port to connect to and credentials for authentication.

dialogs.ssh_host_key

Classes

class king_phisher.client.dialogs.ssh_host_key.**BaseHostKeyDialog**(*application*, *hostname*, *key*)

Bases: *king_phisher.client.gui_utilities.GladeGObject*

A base class for dialogs which show information about SSH host keys. It is assumed that the widgets defined in *dependencies* are present including one button to accept the host key, and one to reject. The class's default response can be set using *default_response*.

__init__(*application*, *hostname*, *key*)

Parameters

- **application** (*KingPhisherClientApplication*) – The application to associate this popup dialog with.
- **hostname** (*str*) – The hostname associated with the key.
- **key** (paramiko.pkey.PKey) – The host's SSH key.

default_response = None

The response that should be selected as the default for the dialog.

class king_phisher.client.dialogs.ssh_host_key.**HostKeyAcceptDialog**(*application*, *hostname*, *key*)

Bases: *king_phisher.client.dialogs.ssh_host_key.BaseHostKeyDialog*

A dialog that shows an SSH host key for a host that has not previously had one associated with it.

class king_phisher.client.dialogs.ssh_host_key.**HostKeyWarnDialog**(*application*, *hostname*, *key*)

Bases: *king_phisher.client.dialogs.ssh_host_key.BaseHostKeyDialog*

A dialog that warns about an SSH host key that does not match the one that was previously stored for the host.

class king_phisher.client.dialogs.ssh_host_key.**MissingHostKeyPolicy**(*application*)

Bases: paramiko.client.MissingHostKeyPolicy

A host key policy for use with paramiko that will validate SSH host keys correctly. If a key is new, the user will be prompted with *HostKeyAcceptDialog* dialog to accept it or if the host key does not match the user will be warned with *HostKeyWarnDialog*. The host keys accepted through this policy are stored in an OpenSSH compatible "known_hosts" file using paramiko.

__init__(*application*)

Parameters **application** (*KingPhisherClientApplication*) – The application which is using this policy.

dialogs.tag_editor

Classes

class king_phisher.client.dialogs.tag_editor.**TagEditorDialog**(**args*, ***kwargs*)

Bases: *king_phisher.client.gui_utilities.GladeGObject*

Display a dialog which can be used to edit the various tags that are present on the remote server. This can be used to rename tags and modify their descriptions.

client.tabs

This package contains modules for providing the content of the top level tabs used by the main application window.

tabs.campaign

This module provides the contents of the tab representing the campaign information in client's graphical interface.

Classes

class king_phisher.client.tabs.campaign.**CampaignViewCredentialsTab**(**args*, ***kwargs*)

Bases: *king_phisher.client.tabs.campaign.CampaignViewGenericTableTab*

Display campaign information regarding submitted credentials.

class king_phisher.client.tabs.campaign.**CampaignViewDashboardTab**(**args*, ***kwargs*)

Bases: *king_phisher.client.tabs.campaign.CampaignViewGenericTab*

Display campaign information on a graphical dash board.

graphs = None
The *CampaignGraph* classes represented on the dash board.

label_text = 'Dashboard'
The tabs label for display in the GUI.

load_campaign_information(*force=True*)
Load the necessary campaign information from the remote server. Unless *force* is True, the *last_load_time* is compared with the *refresh_frequency* to check if the information is stale. If the local data is not stale, this function will return without updating the table.

Parameters **force** (*bool*) – Ignore the load life time and force loading the remote data.

loader_idle_routine()
The routine which refreshes the campaign data at a regular interval.

loader_thread_routine()
The loading routine to be executed within a thread.

class king_phisher.client.tabs.campaign.**CampaignViewDeaddropTab**(**args*, ***kwargs*)

Bases: *king_phisher.client.tabs.campaign.CampaignViewGenericTableTab*

Display campaign information regarding dead drop connections.

class king_phisher.client.tabs.campaign.**CampaignViewGenericTab**(**args*, ***kwargs*)

Bases: *king_phisher.client.gui_utilities.GladeGObject*

This object is meant to be subclassed by all of the tabs which load and display information about the current campaign.

label = None
The Gtk.Label representing this tab with text from *label_text*.

label_text = 'Unknown'
The label of the tab for display in the GUI.

last_load_time = None
The last time the data was loaded from the server.

loader_thread = None
The thread object which loads the data from the server.

loader_thread_lock = None
The `threading.Lock` object used for synchronization between the loader and main threads.

loader_thread_stop = None
The `threading.Event` object used to request that the loader thread stop before completion.

refresh_frequency = None
The lifetime in seconds to wait before refreshing the data from the server.

class king_phisher.client.tabs.campaign.**CampaignViewGenericTableTab**(**args*, ***kwargs*)

Bases: *king_phisher.client.tabs.campaign.CampaignViewGenericTab*

This object is meant to be subclassed by tabs which will display campaign information of different types from specific database tables. The data in this object is refreshed when multiple events occur and it uses an internal timer to represent the last time the data was refreshed.

export_table_to_csv()
Export the data represented by the view to a CSV file.

export_table_to_xlsx_worksheet(*worksheet*, *title_format*)
Export the data represented by the view to an XLSX worksheet.

Parameters

- **worksheet** (`xlsxwriter.worksheet.Worksheet`) – The destination sheet for the store's data.
- **title_format** (`xlsxwriter.format.Format`) – The formatting to use for the title row.

format_cell_data(*cell_data*, *encoding='utf-8'*)
This method provides formatting to the individual cell values returned from the `format_row_data()` function. Values are converted into a format suitable for reading.

Parameters

- **cell** – The value to format.
- **encoding** (*str*) – The encoding to use to coerce the return value into a unicode string.

Returns The formatted cell value.

Return type str

format_node_data(*node*)
This method is overridden by subclasses to format the raw node data returned from the server. The length of the list must equal the number of columns in the table. This method is called for each node in the remote table by the loader thread.

Parameters **node** (*dict*) – The node from a GraphQL query representing data for this table.

Returns The formatted row data.

Return type list

load_campaign_information(*force=True*)
Load the necessary campaign information from the remote server. Unless *force* is True, the *last_load_time* is compared with the *refresh_frequency* to check if the information is stale. If the local data is not stale, this function will return without updating the table.

Parameters **force** (*bool*) – Ignore the load life time and force loading the remote data.

loader_thread_routine(*store*)
The loading routine to be executed within a thread.

Parameters **store** (Gtk.ListStore) – The store object to place the new data.

node_query = None
The GraphQL query used to load a particular node from the remote table. This query is provided with a single parameter of the node's id.

popup_menu = None
The Gtk.Menu object which is displayed when right-clicking in the view area.

table_name = ''
The database table represented by this tab.

table_query = None
The GraphQL query used to load the desired information from the remote table. This query is provided with the following three parameters: campaign, count and cursor.

view_columns = ()
The dictionary map of column numbers to column names starting at column 1.

class king_phisher.client.tabs.campaign.**CampaignViewMessagesTab**(**args*, ***kwargs*)
Bases: *king_phisher.client.tabs.campaign.CampaignViewGenericTableTab*

Display campaign information regarding sent messages.

class king_phisher.client.tabs.campaign.**CampaignViewTab**(*parent*, *application*)
Bases: object

The King Phisher client top-level 'View Campaign' tab. This object manages the sub-tabs which display all the information regarding the current campaign.

__init__(*parent*, *application*)

Parameters

- **parent** (Gtk.Window) – The parent window for this object.
- **application** (Gtk.Application) – The main client application instance.

label = None
The Gtk.Label representing this tabs name.

notebook = None
The Gtk.Notebook for holding sub-tabs.

tabs = None
A dict object holding the sub tabs managed by this object.

class king_phisher.client.tabs.campaign.**CampaignViewVisitsTab**(**args*, ***kwargs*)
Bases: *king_phisher.client.tabs.campaign.CampaignViewGenericTableTab*

Display campaign information regarding incoming visitors.

tabs.mail

This module provides the contents of the tab used to create and send messages as part of a campaign.

Functions

king_phisher.client.tabs.mail.**test_webserver_url**(*target_url*, *secret_id*)
Test the target URL to ensure that it is valid and the server is responding.

Parameters

- **target_url** (*str*) – The URL to make a test request to.
- **secret_id** (*str*) – The King Phisher Server secret id to include in the test request.

Classes

class king_phisher.client.tabs.mail.**MailSenderConfigurationTab**(**args*, ***kwargs*)
Bases: *king_phisher.client.gui_utilities.GladeGObject*

This is the tab which allows the user to configure and set parameters for sending messages as part of a campaign.

label = None
The Gtk.Label representing this tabs name.

class king_phisher.client.tabs.mail.**MailSenderEditTab**(**args*, ***kwargs*)
Bases: *king_phisher.client.gui_utilities.GladeGObject*

This is the tab which adds basic text edition for changing an email template.

label = None
The Gtk.Label representing this tabs name.

load_html_file()
Load the contents of the configured HTML file into the editor.

save_html_file(*force_prompt=False*)
Save the contents from the editor into an HTML file if one is configured otherwise prompt to user to select a file to save as. The user may abort the operation by declining to select a file to save as if they are prompted to do so.

Parameters **force_prompt** – Force prompting the user to select the file to save as.

Return type bool

Returns Whether the contents were saved or not.

show_tab()
Load the message HTML file from disk and configure the tab for editing.

textbuffer = None
The Gtk.TextBuffer used by the :py:attr:textview‘ attribute.

textview = None
The Gtk.TextView object of the editor.

class king_phisher.client.tabs.mail.**MailSenderPreviewTab**(*application*)
Bases: object

This tab uses the WebKit engine to render the HTML of an email so it can be previewed before it is sent.

__init__(*application*)

Parameters **application** (*KingPhisherClientApplication*) – The application instance.

label = None
The Gtk.Label representing this tabs name.

load_html_file()
Load the configured HTML file into the WebKit engine so the contents can be previewed.

show_tab()
Configure the webview to preview the the message HTML file.

webview = None
The *WebKitHTMLView* object used to render the message HTML.

class king_phisher.client.tabs.mail.**MailSenderSendTab**(**args*, ***kwargs*)
Bases: *king_phisher.client.gui_utilities.GladeGObject*

This allows the *MailSenderThread* object to be managed by the user through the GUI. These two classes are very interdependent

label = None
The Gtk.Label representing this tabs name.

notify_sent(*emails_done*, *emails_total*)
A call back use by *MailSenderThread* to notify when an email has been successfully sent to the SMTP server.

Parameters

- **emails_done** (*int*) – The number of email messages that have been sent.
- **emails_total** (*int*) – The total number of email messages that need to be sent.

notify_status(*message*)
A call back use by *MailSenderThread* to update general status information.

Parameters **message** (*str*) – The status message.

notify_stopped()
A callback used by *MailSenderThread* to notify when the thread has stopped.

progressbar = None
The Gtk.ProgressBar instance which is used to display progress of sending messages.

sender_start_failure(*message=None*, *text=None*, *retry=False*)
Handle a failure in starting the message sender thread and perform any necessary clean up.

Parameters

- **message** (*text*) – A message to shown in an error popup dialog.
- **message** – A message to be inserted into the text buffer.
- **retry** (*bool*) – The operation will be attempted again.

sender_thread = None
The *MailSenderThread* instance that is being used to send messages.

text_insert(*message*)
Insert text into the *textbuffer*.

Parameters **message** (*str*) – The text to insert.

textbuffer = None
The Gtk.TextBuffer instance associated with *textview*.

textview = None
The Gtk.TextView object that renders text status messages.

class king_phisher.client.tabs.mail.**MailSenderTab**(*parent*, *application*)
Bases: GObject.GObject

The King Phisher client top-level 'Send Messages' tab. This object manages the sub-tabs which display useful information for configuring, previewing and sending messages as part of a campaign.

GObject Signals *Mail Tab Signals*

__init__(*parent*, *application*)

Parameters

- **parent** (Gtk.Window) – The parent window for this object.
- **application** (Gtk.Application) – The main client application instance.

export_message_data(*path=None*)
Gather and prepare the components of the mailer tab to be exported into a King Phisher message (KPM) archive file suitable for restoring at a later point in time. If *path* is not specified, the user will be prompted to select one and failure to do so will prevent the message data from being exported. This function wraps the emission of the message-data-export signal.

Parameters **path** (*str*) – An optional path of where to save the archive file to.

Returns Whether or not the message archive file was written to disk.

Return type bool

import_message_data()
Process a previously exported message archive file and restore the message data, settings, and applicable files from it. This function wraps the emission of the message-data-import signal.

Returns Whether or not the message archive file was loaded from disk.

Return type bool

label = None
The Gtk.Label representing this tabs name.

notebook = None
The Gtk.Notebook for holding sub-tabs.

tabs = None
A dict object holding the sub tabs managed by this object.

client.widget

widget.extras

This module contains miscellaneous extra GTK widgets.

Classes

class king_phisher.client.widget.extras.**CellRendererBytes**
Bases: Gtk.CellRendererText

A custom Gtk.CellRendererText to render numeric values representing bytes.

class king_phisher.client.widget.extras.**FileChooserDialog**(*title*, *parent*, ***kwargs*)
Bases: Gtk.FileChooserDialog

Display a file chooser dialog with additional convenience methods.

__init__(*title*, *parent*, ***kwargs*)

Parameters

- **title** (*str*) – The title for the file chooser dialog.
- **parent** (Gtk.Window) – The parent window for the dialog.

quick_add_filter(*name*, *patterns*)
Add a filter for displaying files, this is useful in conjunction with *run_quick_open()*.

Parameters

- **name** (*str*) – The name of the filter.
- **patterns** (*list, str*) – The pattern(s) to match.

run_quick_open()
Display a dialog asking a user which file should be opened. The value of target_path in the returned dictionary is an absolute path.

Returns A dictionary with target_uri and target_path keys representing the path chosen.

Return type dict

run_quick_save(*current_name=None*)
Display a dialog which asks the user where a file should be saved. The value of target_path in the returned dictionary is an absolute path.

Parameters **current_name** (*set*) – The name of the file to save.

Returns A dictionary with target_uri and target_path keys representing the path choosen.

Return type dict

run_quick_select_directory()
Display a dialog which asks the user to select a directory to use. The value of target_path in the returned dictionary is an absolute path.

Returns A dictionary with target_uri and target_path keys representing the path chosen.

Return type dict

class king_phisher.client.widget.extras.**WebKitHTMLView**
Bases: WebKitX.WebView

A WebView widget with additional convenience methods for rendering simple HTML content from either files or strings. If a link is opened within the document, the webview will emit the 'open-uri' signal instead of navigating to it.

__init__()

do_open_remote_uri(*uri*, *decision*)

load_html_data(*html_data*, *html_file_uri=None*)
Load arbitrary HTML data into the WebKit engine to be rendered.

Parameters

- **html_data** (*str*) – The HTML data to load into WebKit.
- **html_file_uri** (*str*) – The URI of the file where the HTML data came from.

load_html_file(*html_file*)

signal_button_pressed(*_*, *event*)

signal_decide_policy(*_*, *decision*, *decision_type*)

signal_decide_policy_webkit(*view*, *frame*, *request*, *action*, *policy*)

widget.managers

This module contains classes used for high level management of some GTK widgets.

Classes

class king_phisher.client.widget.managers.**RadioButtonGroupManager**(*glade_gobject*, *button_group_name*)

Bases: object

Manage a group of Gtk.RadioButton objects together to allow the active one to be easily set and identified. The buttons are retrieved from a *GladeGObject* instance and must be correctly named in the *dependencies* attribute as 'radiobutton_group_name_button_name'.

__init__(*glade_gobject*, *button_group_name*)

Parameters

- **glade_gobject** (*GladeGObject*) – The gobject which has the radio buttons set.
- **button_group_name** (*str*) – The name of the group of buttons.

get_active()

Return the name of the active button if one in the group is active. If no button in the group is active, None is returned.

Returns The name of the active button.

Return type str

set_active(*button*)

Set a button in the group as active.

Parameters **button** (*str*) – The name of the button to set as active.

class king_phisher.client.widget.managers.**TreeViewManager**(*treeview*, *selection_mode=None*, *cb_delete=None*, *cb_refresh=None*)

Bases: object

A class that wraps Gtk.TreeView objects that use *Gtk.ListStore* models with additional functions for conveniently displaying text data.

If *cb_delete* is specified, the callback will be called with the treeview instance, and the selection as the parameters.

If *cb_refresh* is specified, the callback will be called without any parameters.

__init__(*treeview*, *selection_mode=None*, *cb_delete=None*, *cb_refresh=None*)

Parameters

- **treeview** (Gtk.TreeView) – The treeview to wrap and manage.

- **selection_mode** (Gtk.SelectionMode) – The selection mode to set for the treeview.
- **cb_delete** (*function*) – An optional callback that can be used to delete entries.

cb_delete = None
An optional callback for deleting entries from the treeview's model.

cb_refresh = None
An optional callback for refreshing the data in the treeview's model.

column_titles = None
An ordered dictionary of storage data columns keyed by their respective column titles.

column_views = None
A dictionary of column treeview's keyed by their column titles.

get_popup_copy_submenu()
Create a Gtk.Menu with entries for copying cell data from the treeview.

Returns The populated copy popup menu.

Return type Gtk.Menu

get_popup_menu(*handle_button_press=True*)
Create a Gtk.Menu with entries for copying and optionally delete cell data from within the treeview. The delete option will only be available if a delete callback was previously set.

Parameters **handle_button_press** (*bool*) – Whether or not to connect a handler for displaying the popup menu.

Returns The populated popup menu.

Return type Gtk.Menu

set_column_color(*background=None*, *foreground=None*, *column_titles=None*)
Set a column in the model to be used as either the background or foreground RGBA color for a cell.

Parameters

- **background** (*int*) – The column id of the model to use as the background color.
- **foreground** (*int*) – The column id of the model to use as the foreground color.
- **column_titles** (*str, tuple*) – The columns to set the color for, if None is specified all columns will be set.

set_column_titles(*column_titles*, *column_offset=0*, *renderers=None*)
Populate the column names of a GTK TreeView and set their sort IDs. This also populates the *column_titles* attribute.

Parameters

- **column_titles** (*list*) – The titles of the columns.
- **column_offset** (*int*) – The offset to start setting column names at.
- **renderers** (*list*) – A list containing custom renderers to use for each column.

Returns A dict of all the Gtk.TreeViewColumn objects keyed by their column id.

Return type dict

treeview = None
The Gtk.TreeView instance being managed.

widget.resources

This module contains resources useful to GTK widgets.

Data

king_phisher.client.widget.resources.**font_desc_italic**
A fake object used to replace missing imports when generating documentation.

king_phisher.client.widget.resources.**renderer_text_desc**
A fake object used to replace missing imports when generating documentation.

Classes

class king_phisher.client.widget.resources.**CompanyEditorGrid**(*destination*)
Bases: *king_phisher.client.gui_utilities.GladeProxy*

An embeddable widget which contains the necessary widgets to edit the various fields of a company object.

children = ('combobox_company_industry', 'entry_company_industry', 'entry_company_name', 'entry_company_des
The children widgets that can be used to edit the fields of the comapny.

name = 'CompanyEditorGrid'
The name of the top level widget in the GTK Builder data file.

widget.completion_providers

This module contains classes for custom auto completion for GtkSourceCompletion. It provides support to recognize special characters and suggests syntax completion.

Functions

king_phisher.client.widget.completion_providers.**get_proposal_terms**(*search*, *tokens*)
Used to iterate through the *search* dictionary definition representing tokens for completion. Terms within this dictionary have a hierarchy to their definition in which keys are always terms represented as strings and values are either sub-dictionaries following the same pattern or None in the case that the term is a leaf node.

Parameters

- **search** (*dict*) – The dictionary to iterate through looking for proposals.
- **tokens** (*list, str*) – List of tokens split on the hierarchy delimiter.

Returns A list of strings to be used for completion proposals.

Return type list

Classes

king_phisher.client.widget.completion_providers.**CustomCompletionProviderBase**
alias of <king_phisher.utilities.Mock object at 0x7f830c76d590>

king_phisher.client.widget.completion_providers.**HTMLComletionProvider**
alias of <king_phisher.utilities.Mock object at 0x7f830bcba390>

king_phisher.client.widget.completion_providers.**JinjaComletionProvider**
alias of <king_phisher.utilities.Mock object at 0x7f830bcba3d0>

king_phisher.client.widget.completion_providers.**JinjaEmailCompletionProvider**
alias of <king_phisher.utilities.Mock object at 0x7f830bcbab90>

client.windows

This package contains modules for providing GTK Window objects used by the client application.

client.windows.campaign_import

This module provides the window through which the user can import King Phisher campaigns from xml files previously exported with the *export* module.

Classes

class king_phisher.client.windows.campaign_import.**ImportCampaignWindow**(**args*, ***kwargs*)

Bases: *king_phisher.client.gui_utilities.GladeGObject*

Display a dialog which allows a new campaign to be created or an existing campaign to be opened.

preprep_xml_data()
This function provides the actions required to see if required IDs are already in the database. If they are not it will clear them out and set subelement.attrib['type'] to null. If the element is required it will set it to a default value. This will normalize the data and ready it for import into the database.

remove_import_campaign()
Used to delete the imported campaign on failure or early exit of the import window, if the user selects to have it removed.

select_xml_campaign()
Prompts the user with a file dialog window to select the King Phisher Campaign XML file to import. Validates the file to make sure it is a Campaign exported from King Phisher and is the correct version to import.

signal_entry_change(_)
When there is a change in the campaign entry field it will check to see if the name is already in use. If it is not in use it will change the sensitivity of the button_import_campaign to allow the user to start the import process.

signal_import_button(_)
This will check to see if the campaign information is present. If campaign information is present it will launch an py:class:*ImportThread* to import the campaign in the background, freeing up the GUI for the user to conduct other functions.

signal_window_delete_event(_, *event*)
Checks to make sure the import campaign thread is closed before closing the window.

client.windows.compare_campaigns

This module provides the window through which the user can compare campaigns across multiple data points in graph format

Classes

class king_phisher.client.windows.compare_campaigns.**CampaignCompWindow**(**args*, ***kwargs*)

Bases: *king_phisher.client.gui_utilities.GladeGObject*

The window which allows the user to select campaigns and compare the data using graphical representation.

load_campaigns()

Load campaigns from the remote server and populate the Gtk.TreeView.

client.windows.main

This module provides the main window used by the client application.

Classes

class king_phisher.client.windows.main.**MainAppWindow**(*config*, *application*)

Bases: Gtk.ApplicationWindow

This is the top level King Phisher client window. This is also the parent window for most GTK objects.

__init__(*config*, *application*)

Parameters

- **config** (*dict*) – The main King Phisher client configuration.
- **application** (*KingPhisherClientApplication*) – The application instance to which this window belongs.

config = None

The main King Phisher client configuration.

export_campaign_visit_geojson()

Export the current campaign visit information to a GeoJSON data file.

export_campaign_xlsx()

Export the current campaign to an Excel compatible XLSX workbook.

export_campaign_xml()

Export the current campaign to an XML data file.

notebook = None

The primary Gtk.Notebook that holds the top level taps of the client GUI.

rpc = None

The *KingPhisherRPCClient* instance.

class king_phisher.client.windows.main.**MainMenuBar**(*application*, *window*)

Bases: *king_phisher.client.gui_utilities.GladeGObject*

The main menu bar for the primary application window. This configures any optional menu items as well as handles all the menu item signals appropriately.

client.windows.plugin_manager

This module provides the window through which the user can enable and disable plugins.

Classes

class king_phisher.client.windows.plugin_manager.**PluginManagerWindow**(**args*, ***kwargs*)

Bases: *king_phisher.client.gui_utilities.GladeGObject*

The window which allows the user to selectively enable and disable plugins for the client application. This also handles configuration changes, so the enabled plugins will persist across application runs.

load_plugins()

Load the plugins which are available into the treeview to make them visible to the user.

client.windows.rpc_terminal

This module provides the RPC Terminal window used by the client application to give the user raw access to the RPC interface.

Data

king_phisher.client.windows.rpc_terminal.**has_vte = True**

Whether the Vte module is available or not.

Classes

class king_phisher.client.windows.rpc_terminal.**RPCTerminal**(*application*)

Bases: object

A terminal using VTE that allows raw RPC methods to be called from within the King Phisher client. This is primarily useful for unofficial and advanced features or debugging and development.

__init__(*application*)

Parameters **application** (*KingPhisherClientApplication*) – The application instance to which this window belongs.

client.application

This module provides the top level GTK application object representing the client application.

Data

king_phisher.client.application.**GTK3_DEFAULT_THEME = 'Adwaita'**

The default GTK3 Theme for style information.

king_phisher.client.application.**USER_DATA_PATH = 'king-phisher'**

The default folder location of user specific data storage.

Classes

class king_phisher.client.application.**KingPhisherClientApplication**(*config_file=None*, *use_plugins=True*, *use_style=True*)

Bases: Gtk.Application

This is the top level King Phisher client object. It contains the custom GObject signals, keeps all the GUI references, and manages the RPC client object. This is also the parent window for most GTK objects.

GObject Signals *Application Signals*

add_reference(*ref_object*)
Add *ref_object* to the *references* so the object won't be garbage collected. The object must either be a *GladeGObject* or Gtk.Widget instance so a cleanup function can be attached to a destroy signal to remove the reference automatically.

Parameters **ref_object** (*GladeGObject*, Gtk.Widget) – The object to store a reference to.

campaign_rename()
Show a dialog prompting the user to for the a new name to assign to the currently selected campaign.

config = None
The primary King Phisher client configuration.

config_file = None
The file containing the King Phisher client configuration.

do_campaign_delete(*campaign_id*)
Delete the campaign on the server. A confirmation dialog will be displayed before the operation is performed. If the campaign is deleted and a new campaign is not selected with *show_campaign_selection()*, the client will quit.

do_config_load(*load_defaults*)
Load the client configuration from disk and set the *config* attribute.

Parameters **load_defaults** (*bool*) – Load missing options from the template configuration file.

do_server_disconnected()
Clean up the connections to the server and disconnect. This logs out of the RPC, closes the server event socket, and stops the SSH forwarder.

do_sftp_client_start()
Start the client's preferred sftp client application in a new process.

load_server_config()
Load the necessary values from the server's configuration.

main_window = None
The primary top-level *MainAppWindow* instance.

merge_config(*config_file*, *strict=True*)
Merge the configuration information from the specified configuration file. Only keys which exist in the currently loaded configuration are copied over while non-existent keys are skipped. The contents of the new configuration overwrites the existing.

Parameters

- **strict** (*bool*) – Do not try remove trailing commas from the JSON data.
- **config_file** (*str*) – The path to the configuration file to merge.

quit (*optional=False*)
Quit the client and perform any necessary clean up operations. If *optional* is False then the exit-confirm signal will not be sent and there will not be any opportunities for the client to cancel the operation.

Parameters **optional** (*bool*) – Whether the quit is request is optional or not.

references = None
A list to store references to arbitrary objects in for avoiding garbage collection.

rpc = None
The *KingPhisherRPCClient* instance for the application.

server_events = None
The *ServerEventSubscriber* instance for the application to receive server events.

show_campaign_graph (*graph_name*)
Create a new *CampaignGraph* instance and make it into a window. *graph_name* must be the name of a valid, exported graph provider.

Parameters **graph_name** (*str*) – The name of the graph to make a window of.

show_campaign_selection ()
Display the campaign selection dialog in a new *CampaignSelectionDialog* instance.

Returns Whether or not a campaign was selected.

Return type bool

show_preferences ()
Display a *dialogs.configuration.ConfigurationDialog* instance and saves the configuration to disk if cancel is not selected.

stop_remote_service ()
Stop the remote King Phisher server. This will request that the server stop processing new requests and exit. This will display a confirmation dialog before performing the operation. If the remote service is stopped, the client will quit.

client.client_rpc

This module facilitates communication with the server application over the RPC interface.

Data

king_phisher.client.client_rpc.**UNRESOLVED** = UNRESOLVED

Functions

king_phisher.client.client_rpc.**vte_child_routine** (*config*)
This is the method which is executed within the child process spawned by VTE. It expects additional values to be set in the *config* object so it can initialize a new *KingPhisherRPCClient* instance. It will then drop into an interpreter where the user may directly interact with the rpc object.

Parameters **config** (*str*) – A JSON encoded client configuration.

Classes

class king_phisher.client.client_rpc.**KingPhisherRPCClient**(**args*, ***kwargs*)
Bases: advancedhttpserver.RPCClientCached

The main RPC object for communicating with the King Phisher Server over RPC.

geoip_lookup(*ip*)
Look up the geographic location information for the specified IP address in the server's geoip database.

Parameters **ip** (ipaddress.IPv4Address, str) – The IP address to lookup.

Returns The geographic location information for the specified IP address.

Return type *GeoLocation*

geoip_lookup_multi(*ips*)
Look up the geographic location information for the specified IP addresses in the server's geoip database. Because results are cached for optimal performance, IP addresses to be queried should be grouped and sorted in a way that is unlikely to change, i.e. by a timestamp.

Parameters **ips** (*list, set, tuple*) – The IP addresses to lookup.

Returns The geographic location information for the specified IP address.

Return type dict

get_tag_model(*tag_table*, *model=None*)
Load tag information from a remote table into a Gtk.ListStore instance. Tables compatible with the tag interface must have id, name and description fields. If no *model* is provided a new one will be created, else the current model will be cleared.

Parameters

- **tag_table** (*str*) – The name of the table to load tag information from.
- **model** (Gtk.ListStore) – The model to place the information into.

Returns The model with the loaded data from the server.

Return type Gtk.ListStore

graphql(*query*, *query_vars=None*)

login(*username*, *password*, *otp=None*)
Authenticate to the remote server. This is required before calling RPC methods which require an authenticated session.

Parameters

- **username** (*str*) – The username to authenticate with.
- **password** (*str*) – The password to authenticate with.
- **otp** (*str*) – An optional one time password as a 6 digit string to provide if the account requires it.

Returns The login result and an accompanying reason.

Return type tuple

ping()
Call the ping RPC method on the remote server to ensure that it is responsive. On success this method will always return True, otherwise an exception will be thrown.

Returns True

Return type bool

reconnect()
Reconnect to the remote server.

remote_row_resolve(*row*)
Take a *RemoteRow* instance and load all fields which are *UNRESOLVED*. If all fields are present, no modifications are made.

Parameters **row** – The row who's data is to be resolved.

Return type *RemoteRow*

Returns The row with all of it's fields fully resolved.

Return type *RemoteRow*

remote_table(*table*, *query_filter=None*)
Iterate over a remote database table hosted on the server. Rows are yielded as named tuples whose fields are the columns of the specified table.

Parameters **table** (*str*) – The table name to retrieve.

Returns A generator which yields rows of named tuples.

Return type tuple

remote_table_row(*table*, *row_id*, *cache=False*, *refresh=False*)
Get a row from the specified table by it's id, optionally caching it.

Parameters

- **table** (*str*) – The table in which the row exists.
- **row_id** – The value of the row's id column.
- **cache** (*bool*) – Whether to use the cache for this row.
- **refresh** (*bool*) – If *cache* is True, get the current row value and store it.

Returns The remote row as a named tuple of the specified table.

Return type tuple

class king_phisher.client.client_rpc.**RemoteRow**(*rpc*, **args*, ***kwargs*)
Bases: king_phisher.client.client_rpc._RemoteRow

A generic class representing a row of data from the remote King Phisher server.

commit()
Send this object to the server to update the remote instance.

client.export

This module provides functionality for exporting information from the client application into a variety of formats.

Functions

king_phisher.client.export.**campaign_to_xml**(*rpc*, *campaign_id*, *xml_file*, *encoding='utf-8'*)
Load all information for a particular campaign and dump it to an XML file.

Parameters

- **rpc** (*KingPhisherRPCClient*) – The connected RPC instance to load the information with.
- **campaign_id** – The ID of the campaign to load the information for.
- **xml_file** (*str*) – The destination file for the XML data.
- **encoding** (*str*) – The encoding to use for strings.

king_phisher.client.export.**campaign_visits_to_geojson**(*rpc*, *campaign_id*, *geojson_file*)

Export the geo location information for all the visits of a campaign into the GeoJSON format.

Parameters

- **rpc** (*KingPhisherRPCClient*) – The connected RPC instance to load the information with.
- **campaign_id** – The ID of the campaign to load the information for.
- **geojson_file** (*str*) – The destination file for the GeoJSON data.

king_phisher.client.export.**convert_value**(*table_name*, *key*, *value*)

Perform any conversions necessary to neatly display the data in XML format.

Parameters

- **table_name** (*str*) – The table name that the key and value pair are from.
- **key** (*str*) – The data key.
- **value** – The data value to convert.

Returns The converted value.

Return type str

king_phisher.client.export.**message_data_to_kpm**(*message_config*, *target_file*, *encoding='utf-8'*)

Save details describing a message to the target file.

Parameters

- **message_config** (*dict*) – The message details from the client configuration.
- **target_file** (*str*) – The file to write the data to.
- **encoding** (*str*) – The encoding to use for strings.

king_phisher.client.export.**liststore_export**(*store*, *columns*, *cb_write*, *cb_write_args*, *row_offset=0*, *write_columns=True*)

A function to facilitate writing values from a list store to an arbitrary callback for exporting to different formats. The callback will be called with the row number, the column values and the additional arguments specified in **cb_write_args*.

```
cb_write(row, column_values, *cb_write_args).
```

Parameters

- **store** (*Gtk.ListStore*) – The store to export the information from.
- **columns** (*dict*) – A dictionary mapping store column ids to the value names.
- **cb_write** (*function*) – The callback function to be called for each row of data.
- **cb_write_args** (*tuple*) – Additional arguments to pass to *cb_write*.

- **row_offset** (*int*) – A modifier value to add to the row numbers passed to *cb_write*.
- **write_columns** (*bool*) – Write the column names to the export.

Returns The number of rows that were written.

Return type int

king_phisher.client.export.**liststore_to_csv**(*store*, *target_file*, *columns*)
Write the contents of a Gtk.ListStore to a csv file.

Parameters

- **store** (Gtk.ListStore) – The store to export the information from.
- **target_file** (*str*) – The destination file for the CSV data.
- **columns** (*dict*) – A dictionary mapping store column ids to the value names.

Returns The number of rows that were written.

Return type int

king_phisher.client.export.**liststore_to_xlsx_worksheet**(*store*, *worksheet*, *columns*, *title_format*, *xlsx_options=None*)
Write the contents of a Gtk.ListStore to an XLSX workseet.

Parameters

- **store** (Gtk.ListStore) – The store to export the information from.
- **worksheet** (xlsxwriter.worksheet.Worksheet) – The destination sheet for the store's data.
- **columns** (*dict*) – A dictionary mapping store column ids to the value names.
- **xlsx_options** (XLSXWorksheetOptions) – A collection of additional options for formatting the Excel Worksheet.

Returns The number of rows that were written.

Return type int

client.graphs

This module provides the functionality to support the client application's graphing capabilities.

Data

king_phisher.client.graphs.**has_matplotlib = False**
Whether the matplotlib module is available.

king_phisher.client.graphs.**has_matplotlib_basemap = False**
Whether the mpl_toolkits.basemap module is available.

Functions

king_phisher.client.graphs.**export_graph_provider**(*cls*)
Decorator to mark classes as valid graph providers. This decorator also sets the name attribute.

Parameters **cls** (*class*) – The class to mark as a graph provider.

Returns The *cls* parameter is returned.

king_phisher.client.graphs.**get_graph**(*graph_name*)
Return the graph providing class for *graph_name*. The class providing the specified graph must have been previously exported using *export_graph_provider()*.

Parameters **graph_name** (*str*) – The name of the graph provider.

Returns The graph provider class.

Return type *CampaignGraph*

king_phisher.client.graphs.**get_graphs**()
Get a list of all registered graph providers.

Returns All registered graph providers.

Return type list

Classes

class king_phisher.client.graphs.**GraphBase**(*application*, *size_request=None*, *style_context=None*)
Bases: object

A basic graph provider for using matplotlib to create graph representations of campaign data. This class is meant to be subclassed by real providers.

__init__(*application*, *size_request=None*, *style_context=None*)

Parameters **size_request** (*tuple*) – The size to set for the canvas.

config = None
A reference to the King Phisher client configuration.

get_color(*color_name*, *default*)
Get a color by its style name such as 'fg' for foreground. If the specified color does not exist, default will be returned. The underlying logic for this function is provided by *gtk_style_context_get_color()*.

Parameters

- **color_name** (*str*) – The style name of the color.
- **default** – The default color to return if the specified one was not found.

Returns The desired color if it was found.

Return type tuple

graph_title = 'Unknown'
The title that will be given to the graph.

make_window()
Create a window from the figure manager.

Returns The graph in a new, dedicated window.

Return type Gtk.Window

minimum_size = None
An absolute minimum size for the canvas.

name = 'Unknown'
The name of the graph provider.

name_human = 'Unknown'
The human readable name of the graph provider used for UI identification.

resize(*width=0*, *height=0*)
Attempt to resize the canvas. Regardless of the parameters the canvas will never be resized to be smaller than *minimum_size*.

Parameters

- **width** (*int*) – The desired width of the canvas.
- **height** (*int*) – The desired height of the canvas.

table_subscriptions = []
A list of tables from which information is needed to produce the graph.

class king_phisher.client.graphs.**CampaignGraph**(*application*, *size_request=None*, *style_context=None*)
Bases: *king_phisher.client.graphs.GraphBase*

Graph format used for the graphs generated in the dashboard and in the create graphs tab.

load_graph()
Load the graph information via *refresh()*.

refresh(*info_cache=None*, *stop_event=None*)
Refresh the graph data by retrieving the information from the remote server.

Parameters

- **info_cache** (*dict*) – An optional cache of data tables.
- **stop_event** (threading.Event) – An optional object indicating that the operation should stop.

Returns A dictionary of cached tables from the server.

Return type dict

class king_phisher.client.graphs.**CampaignGraphMessageResults**(**args*, ***kwargs*)
Bases: king_phisher.client.graphs.CampaignPieGraph

Display the percentage of messages which resulted in a visit.

graph_title = 'Campaign Message Results'

name = 'MessageResults'

name_human = 'Pie - Message Results'

table_subscriptions = ('credentials', 'visits')

class king_phisher.client.graphs.**CampaignGraphOverview**(**args*, ***kwargs*)
Bases: king_phisher.client.graphs.CampaignBarGraph

Display a graph which represents an overview of the campaign.

graph_title = 'Campaign Overview'

name = 'Overview'

name_human = 'Bar - Campaign Overview'

table_subscriptions = ('credentials', 'visits')

class king_phisher.client.graphs.**CampaignGraphPasswordComplexityPie**(**args*, ***kwargs*)

Bases: king_phisher.client.graphs.CampaignPieGraph

Display a graph which displays the number of passwords which meet standard complexity requirements.

graph_title = 'Campaign Password Complexity'

name = 'PasswordComplexityPie'

name_human = 'Pie - Password Complexity'

table_subscriptions = ('credentials',)

class king_phisher.client.graphs.**CampaignGraphVisitorInfo**(**args*, ***kwargs*)

Bases: king_phisher.client.graphs.CampaignBarGraph

Display a graph which shows the different operating systems seen from visitors.

graph_title = 'Campaign Visitor OS Information'

name = 'VisitorInfo'

name_human = 'Bar - Visitor OS Information'

table_subscriptions = ('visits',)

class king_phisher.client.graphs.**CampaignGraphVisitorInfoPie**(**args*, ***kwargs*)

Bases: king_phisher.client.graphs.CampaignPieGraph

Display a graph which compares the different operating systems seen from visitors.

graph_title = 'Campaign Visitor OS Information'

name = 'VisitorInfoPie'

name_human = 'Pie - Visitor OS Information'

table_subscriptions = ('visits',)

class king_phisher.client.graphs.**CampaignGraphVisitsMap**(*application*, *size_request=None*, *style_context=None*)

Bases: *king_phisher.client.graphs.CampaignGraph*

A base class to display a map which shows the locations of visit origins.

color_with_creds

color_without_creds

draw_states = False

graph_title = 'Campaign Visit Locations'

is_available = False

table_subscriptions = ('credentials', 'visits')

king_phisher.client.graphs.**CampaignGraphVisitsMapUSA**

king_phisher.client.graphs.**CampaignGraphVisitsMapWorld**

class king_phisher.client.graphs.**CampaignGraphVisitsTimeline**(**args*, ***kwargs*)

Bases: king_phisher.client.graphs.CampaignLineGraph

Display a graph which represents the visits of a campaign over time.

graph_title = 'Campaign Visits Timeline'

name = 'VisitsTimeline'

name_human = 'Line - Visits Timeline'

table_subscriptions = ('visits',)

class king_phisher.client.graphs.**CampaignCompGraph**(**args*, ***kwargs*)
Bases: *king_phisher.client.graphs.GraphBase*

Display selected campaigns data by order of campaign start date.

load_graph(*campaigns*)
Load the information to compare the specified and paint it to the canvas. Campaigns are graphed on the X-axis in the order that they are provided. No sorting of campaigns is done by this method.

Parameters **campaigns** (*tuple*) – A tuple containing campaign IDs to compare.

client.gui_utilities

This module provides various utility functions specific to the graphical nature of the client application.

Data

king_phisher.client.gui_utilities.**GOBJECT_PROPERTY_MAP**
The dictionary which maps GObjects to either the names of properties to store text or a tuple which contains a set and get function. If a tuple of two functions is specified the set function will be provided two parameters, the object and the value and the get function will just be provided the object.

Functions

king_phisher.client.gui_utilities.**glib_idle_add_wait**(*function*, **args*)
Execute *function* in the main GTK loop using GLib.idle_add() and block until it has completed. This is useful for threads that need to update GUI data.

Parameters

- **function** (*function*) – The function to call.
- **args** – The arguments to *function*.

Returns The result of the function call.

king_phisher.client.gui_utilities.**gobject_get_value**(*gobject*, *gtype=None*)
Retreive the value of a GObject widget. Only objects with value retrieving functions present in the *GOBJECT_PROPERTY_MAP* can be processed by this function.

Parameters

- **gobject** (GObject.Object) – The object to retrieve the value for.
- **gtype** (*str*) – An explicit type to treat *gobject* as.

Returns The value of *gobject*.

Return type str

king_phisher.client.gui_utilities.**gobject_signal_accumulator**(*test=None*)
Create an accumulator function for use with GObject signals. All return values will be collected and returned in a list. If provided, *test* is a callback that will be called with two arguments, the return value from the handler and the list of accumulated return values.

```
stop = test(retval, accumulated)
```

Parameters **test** – A callback to test whether additional handler should be executed.

king_phisher.client.gui_utilities.**gobject_signal_blocked**(**args*, ***kwds*)
This is a context manager that can be used with the 'with' statement to execute a block of code while *signal_name* is blocked.

Parameters

- **gobject** (GObject.Object) – The object to block the signal on.
- **signal_name** (*str*) – The name of the signal to block.

king_phisher.client.gui_utilities.**gtk_calendar_get_pydate**(*calendar*)
Get the Python date from a Gtk.Calendar instance.

Parameters **calendar** (Gtk.Calendar) – The calendar to get the date from.

Returns The date as returned by the calendar's get_date() method.

Return type datetime.date

king_phisher.client.gui_utilities.**gtk_calendar_set_pydate**(*calendar*, *pydate*)
Set the date on a Gtk.Calendar instance from a Python datetime.date object.

Parameters

- **calendar** (Gtk.Calendar) – The calendar to set the date for.
- **pydate** (datetime.date) – The date to set on the calendar.

king_phisher.client.gui_utilities.**gtk_list_store_search**(*list_store*, *value*, *column=0*)
Search a Gtk.ListStore for a value and return a Gtk.TreeIter to the first match.

Parameters

- **list_store** (Gtk.ListStore) – The list store to search.
- **value** – The value to search for.
- **column** (*int*) – The column in the row to check.

Returns The row on which the value was found.

Return type Gtk.TreeIter

king_phisher.client.gui_utilities.**gtk_menu_get_item_by_label**(*menu*, *label*)
Retrieve a menu item from a menu by it's label. If more than one items share the same label, only the first is returned.

Parameters

- **menu** (Gtk.Menu) – The menu to search for the item in.
- **label** (*str*) – The label to search for in *menu*.

Returns The identified menu item if it could be found, otherwise None is returned.

Return type Gtk.MenuItem

king_phisher.client.gui_utilities.**gtk_menu_insert_by_path**(*menu*, *menu_path*, *menu_item*)
Add a new menu item into the existing menu at the path specified in *menu_path*.

Parameters

- **menu** (Gtk.Menu Gtk.MenuBar) – The existing menu to add the new item to.
- **menu_path** (*list*) – The labels of submenus to traverse to insert the new item.
- **menu_item** (Gtk.MenuItem) – The new menu item to insert.

king_phisher.client.gui_utilities.**gtk_menu_position**(*event*, **args*)

Create a menu at the given location for an event. This function is meant to be used as the *func* parameter for the Gtk.Menu.popup() method. The *event* object must be passed in as the first parameter, which can be accomplished using functools.partial().

Parameters **event** – The event to retrieve the coordinates for.

king_phisher.client.gui_utilities.**gtk_style_context_get_color**(*sc*, *color_name*, *default=None*)

Look up a color by it's name in the Gtk.StyleContext specified in *sc*, and return it as an Gdk.RGBA instance if the color is defined. If the color is not found, *default* will be returned.

Parameters

- **sc** (Gtk.StyleContext) – The style context to use.
- **color_name** (*str*) – The name of the color to lookup.
- **default** (str, Gdk.RGBA) – The default color to return if the specified color was not found.

Returns The color as an RGBA instance.

Return type Gdk.RGBA

king_phisher.client.gui_utilities.**gtk_sync**()

Wait while all pending GTK events are processed.

king_phisher.client.gui_utilities.**gtk_treesortable_sort_func_numeric**(*model*, *iter1*, *iter2*, *column_id*)

Sort the model by comparing text numeric values with place holders such as 1,337. This is meant to be set as a sorting function using Gtk.TreeSortable.set_sort_func(). The user_data parameter must be the column id which contains the numeric values to be sorted.

Parameters

- **model** (Gtk.TreeSortable) – The model that is being sorted.
- **iter1** (Gtk.TreeIter) – The iterator of the first item to compare.
- **iter2** (Gtk.TreeIter) – The iterator of the second item to compare.
- **column_id** – The ID of the column containing numeric values.

Returns An integer, -1 if item1 should come before item2, 0 if they are the same and 1 if item1 should come after item2.

Return type int

king_phisher.client.gui_utilities.**gtk_treeview_selection_to_clipboard**(*treeview*, *columns=0*)

Copy the currently selected values from the specified columns in the treeview to the users clipboard. If no value is selected in the treeview, then the clipboard is left unmodified. If multiple values are selected, they will all be placed in the clipboard on separate lines.

Parameters

- **treeview** (Gtk.TreeView) – The treeview instance to get the selection from.
- **column** (*int, list, tuple*) – The column numbers to retrieve the value for.

king_phisher.client.gui_utilities.**gtk_treeview_selection_iterate**(*treeview*)
Iterate over the a treeview's selected rows.

Parameters **treeview** (Gtk.TreeView) – The treeview for which to iterate over.

Returns The rows which are selected within the treeview.

Return type Gtk.TreeIter

king_phisher.client.gui_utilities.**gtk_treeview_set_column_titles**(*treeview*, *column_titles*, *column_offset=0*, *renderers=None*)
Populate the column names of a GTK TreeView and set their sort IDs.

Parameters

- **treeview** (Gtk.TreeView) – The treeview to set column names for.
- **column_titles** (*list*) – The names of the columns.
- **column_offset** (*int*) – The offset to start setting column names at.
- **renderers** (*list*) – A list containing custom renderers to use for each column.

Returns A dict of all the Gtk.TreeViewColumn objects keyed by their column id.

Return type dict

king_phisher.client.gui_utilities.**gtk_widget_destroy_children**(*widget*)
Destroy all GTK child objects of *widget*.

Parameters **widget** (Gtk.Widget) – The widget to destroy all the children of.

king_phisher.client.gui_utilities.**show_dialog**(*message_type*, *message*, *parent*, *secondary_text=None*, *message_buttons=<king_phisher.utilities.MockObject object>*, *use_markup=False*, *secondary_use_markup=False*)
Display a dialog and return the response. The response is dependent on the value of *message_buttons*.

Parameters

- **message_type** (Gtk.MessageType) – The GTK message type to display.
- **message** (*str*) – The text to display in the dialog.
- **parent** (Gtk.Window) – The parent window that the dialog should belong to.
- **secondary_text** (*str*) – Optional subtext for the dialog.
- **message_buttons** (Gtk.ButtonsType) – The buttons to display in the dialog box.
- **use_markup** (*bool*) – Whether or not to treat the message text as markup.
- **secondary_use_markup** (*bool*) – Whether or not to treat the secondary text as markup.

Returns The response of the dialog.

Return type int

king_phisher.client.gui_utilities.**show_dialog_exc_socket_error**(*error*, *parent*, *title=None*)
Display an error dialog with details regarding a `socket.error` exception that has been raised.

Parameters

- **error** (`socket.error`) – The exception instance that has been raised.
- **parent** (`Gtk.Window`) – The parent window that the dialog should belong to.
- **title** – The title of the error dialog that is displayed.

king_phisher.client.gui_utilities.**show_dialog_error**(**args*, ***kwargs*)
Display an error dialog with *show_dialog()*.

king_phisher.client.gui_utilities.**show_dialog_info**(**args*, ***kwargs*)
Display an informational dialog with *show_dialog()*.

king_phisher.client.gui_utilities.**show_dialog_warning**(**args*, ***kwargs*)
Display an warning dialog with *show_dialog()*.

king_phisher.client.gui_utilities.**show_dialog_yes_no**(**args*, ***kwargs*)
Display a dialog which asks a yes or no question with *show_dialog()*.

Returns True if the response is Yes.

Return type bool

king_phisher.client.gui_utilities.**which_glade**()
Locate the glade data file which stores the UI information in a Gtk Builder format.

Returns The path to the glade data file.

Return type str

Classes

class king_phisher.client.gui_utilities.**FileMonitor**(*path*, *on_changed*)
Bases: `object`

Monitor a file for changes.

class king_phisher.client.gui_utilities.**GladeDependencies**(*children=None*, *top_level=None*, *name=None*)
Bases: `object`

A class for defining how objects should be loaded from a GTK Builder data file for use with *GladeGObject*.

__init__(*children=None*, *top_level=None*, *name=None*)

children
A tuple of string names or *GladeProxy* instances listing the children widgets to load from the parent.

name
The string of the name of the top level parent widget to load.

top_level
A tuple of string names listing additional top level widgets to load such as images.

class king_phisher.client.gui_utilities.**GladeGObjectMeta**(**args*, ***kwargs*)
Bases: `type`

A meta class that will update the *GladeDependencies.name* value in the *GladeGObject.dependencies* attribute of instances if no value is defined.

class assigned_name
Bases: str

A type subclassed from str that is used to define names which have been automatically assigned by this class.

class king_phisher.client.gui_utilities.**GladeGObject**(*application*)
Bases: king_phisher.client.gui_utilities._GladeGObject

A base object to wrap GTK widgets loaded from Glade data files. This provides a number of convenience methods for managing the main widget and child widgets. This class is meant to be subclassed by classes representing objects from the Glade data file.

__init__(*application*)

Parameters **application** (Gtk.Application) – The parent application for this object.

application = None
The parent Gtk.Application instance.

config = None
A reference to the King Phisher client configuration.

config_prefix = ''
A prefix to be used for keys when looking up value in the *config*.

dependencies = <GladeDependencies name='GladeGObject' >
A *GladeDependencies* instance which defines information for loading the widget from the GTK builder data.

destroy()
Destroy the top-level GObject.

get_entry_value(*entry_name*)
Get the value of the specified entry then remove leading and trailing white space and finally determine if the string is empty, in which case return None.

Parameters **entry_name** (*str*) – The name of the entry to retrieve text from.

Returns Either the non-empty string or None.

Return type None, str

gobjects = None
A *FreezableDict* which maps gobjects to their unique GTK Builder id.

gtk_builder = None
A Gtk.Builder instance used to load Glade data with.

gtk_builder_get(*gobject_id*, *parent_name=None*)
Find the child GObject with name *gobject_id* from the GTK builder.

Parameters

- **gobject_id** (*str*) – The object name to look for.
- **parent_name** (*str*) – The name of the parent object in the builder data file.

Returns The GObject as found by the GTK builder.

Return type GObject.Object

objects_load_from_config()
Iterate through *gobjects* and set the GObject's value from the corresponding value in the *config*.

objects_persist = True
Whether objects should be automatically loaded from and saved to the configuration.

objects_save_to_config()

parent

top_gobject = 'gobject'
The name of the attribute to set a reference of the top level GObject to.

class king_phisher.client.gui_utilities.**GladeProxy**(*destination*)
Bases: object

A class that can be used to load another top level widget from the GTK builder data file in place of a child. This is useful for reusing small widgets as children in larger ones.

__init__(*destination*)

children = ()
A tuple of string names or *GladeProxy* instances listing the children widgets to load from the top level.

destination
A *GladeProxyDestination* instance describing how this proxied widget should be added to the parent.

name = None
The string of the name of the top level widget to load.

class king_phisher.client.gui_utilities.**GladeProxyDestination**(*widget*, *method*, *args=None*, *kwargs=None*)
Bases: object

A class that is used to define how a *GladeProxy* object shall be loaded into a parent *GladeGObject* instance. This includes the information such as what container widget in the parent the proxied widget should be added to and what method should be used. The proxied widget will be added to the parent by calling *method* with the proxied widget as the first argument.

__init__(*widget*, *method*, *args=None*, *kwargs=None*)

args
Arguments to append after the proxied child instance when calling *method*.

kwargs
Key word arguments to append after the proxied child instance when calling *method*.

method
The method of the parent widget that should be called to add the proxied child.

widget
The name of the parent widget for this proxied child.

client.mailer

This module provides the functionality used to create and sending messages from the client application.

Functions

king_phisher.client.mailer.**guess_smtp_server_address**()

Guess the IP address of the SMTP server that will be connected to given the SMTP host information and an optional SSH forwarding host. If a hostname is in use it will be resolved to an IP address, either IPv4 or IPv6 and in that order. If a hostname resolves to multiple IP addresses, None will be returned. This function is intended to guess the SMTP servers IP address given the client configuration so it can be used for SPF record checks.

Parameters

- **host** (*str*) – The SMTP server that is being connected to.
- **forward_host** (*str*) – An optional host that is being used to tunnel the connection.

Returns The IP address of the SMTP server.

Return type None, ipaddress.IPv4Address, ipaddress.IPv6Address

king_phisher.client.mailer.**render_message_template**(*template*, *config*, *target=None*, *analyze=False*)

Take a message from a template and format it to be sent by replacing variables and processing other template directives. If the *target* parameter is not set, a placeholder will be created and the message will be formatted to be previewed.

Parameters

- **template** (*str*) – The message template.
- **config** (*dict*) – The King Phisher client configuration.
- **target** (*MessageTarget*) – The messages intended target information.
- **analyze** (*bool*) – Set the template environment to analyze mode.

Returns The formatted message.

Return type str

Classes

class king_phisher.client.mailer.**MailSenderThread**(*application*, *target_file*, *rpc*, *tab=None*)

Bases: threading.Thread

The King Phisher threaded email message sender. This object manages the sending of emails for campaigns and supports pausing the sending of messages which can later be resumed by unpausing. This object reports its information to the GUI through an optional *MailSenderSendTab* instance, these two objects are very interdependent.

__init__(*application*, *target_file*, *rpc*, *tab=None*)

Parameters

- **application** (*KingPhisherClientApplication*) – The GTK application that the thread is associated with.
- **target_file** (*str*) – The CSV formatted file to read message targets from.
- **tab** (*MailSenderSendTab*) – The GUI tab to report information to.
- **rpc** (*KingPhisherRPCClient*) – The client's connected RPC instance.

count_messages()
Count the number of targets that will be sent messages.

Returns The number of targets that will be sent messages.

Return type int

create_calendar_invite(*target*, *attachments*)
Create a MIME calendar invite to be sent from a set of parameters.

Parameters

- **target** (*MessageTarget*) – The information for the messages intended recipient.
- **uid** (*str*) – The message's unique identifier.
- **attachments** (Attachments) – The attachments to add to the created message.

Returns The new MIME message.

Return type email.mime.multipart.MIMEMultipart

create_email(*target*, *attachments*)
Create a MIME email to be sent from a set of parameters.

Parameters

- **target** (*MessageTarget*) – The information for the messages intended recipient.
- **uid** (*str*) – The message's unique identifier.
- **attachments** (MessageAttachments) – The attachments to add to the created message.

Returns The new MIME message.

Return type email.mime.multipart.MIMEMultipart

get_mime_attachments()
Return a *MessageAttachments* object containing both the images and raw files to be included in sent messages.

Returns A namedtuple of both files and images in their MIME containers.

Return type *MessageAttachments*

missing_files()
Return a list of all missing or unreadable files which are referenced by the message template.

Returns The list of unusable files.

Return type list

pause()
Sets the *running* and *paused* flags correctly to indicate that the object is paused.

paused = None
A threading.Event object indicating if the email sending operation is or should be paused.

process_pause(*set_pause=False*)
Pause sending emails if a pause request has been set.

Parameters **set_pause** (*bool*) – Whether to request a pause before processing it.

Returns Whether or not the sending operation was cancelled during the pause.

Return type bool

running = None
A `threading.Event` object indicating if emails are being sent.

send_message(*target_email*, *msg*)
Send an email using the connected SMTP server.

Parameters

- **target_email** (*str*) – The email address to send the message to.
- **msg** (`mime.multipart.MIMEMultipart`) – The formatted message to be sent.

server_smtp_connect()
Connect and optionally authenticate to the configured SMTP server.

Returns The connection status as one of the *ConnectionErrorReason* constants.

server_smtp_disconnect()
Clean up and close the connection to the remote SMTP server.

server_smtp_reconnect()
Disconnect from the remote SMTP server and then attempt to open a new connection to it.

Returns The reconnection status.

Return type bool

server_ssh_connect()
Connect to the remote SMTP server over SSH and configure port forwarding with *SSHTCPForwarder* for tunneling SMTP traffic.

Returns The connection status as one of the *ConnectionErrorReason* constants.

smtp_connection = None
The `smtplib.SMTP` connection instance.

stop()
Requests that the email sending operation stop. It can not be resumed from the same position. This function blocks until the stop request has been processed and the thread exits.

tab = None
The optional *MailSenderSendTab* instance for reporting status messages to the GUI.

tab_notify_sent(*emails_done*, *emails_total*)
Notify the tab that messages have been sent.

Parameters

- **emails_done** (*int*) – The number of emails that have been sent.
- **emails_total** (*int*) – The total number of emails that are going to be sent.

tab_notify_status(*message*)
Handle a status message regarding the message sending operation.

Parameters **message** (*str*) – The notification message.

tab_notify_stopped()
Notify the tab that the message sending operation has stopped.

target_file = None
The name of the target file in CSV format.

unpause()
Sets the *running* and *paused* flags correctly to indicate that the object is no longer paused.

class king_phisher.client.mailer.**MessageAttachments** (*files*, *images*)
A named tuple for holding both image and file attachments for a message.

files
Alias for field number 0

images
Alias for field number 1

class king_phisher.client.mailer.**MessageTarget** (*first_name*, *last_name*, *email_address*, *uid*, *department=None*, *line=None*)
A simple class for holding information regarding a messages intended recipient.

department
The target recipient's department name.

email_address
The target recipient's email address.

first_name
The target recipient's first name.

last_name
The target recipient's last name.

line
The line number in the file from which this target was loaded.

uid
The unique identifier that is going to be used for this target.

class king_phisher.client.mailer.**TopMIMEMultipart** (*mime_type*, *config*, *target*)
Bases: email.mime.multipart.MIMEMultipart

A mime.multipart.MIMEMultipart subclass for representing the top / outer most part of a MIME multipart message.

__init__ (*mime_type*, *config*, *target*)

Parameters

- **mime_type** (*str*) – The type of this part such as related or alternative.
- **config** (*dict*) – The client configuration.
- **target** (*MessageTarget*) – The target information for the messages intended recipient.

client.plugins

Classes

class king_phisher.client.plugins.**ClientOptionBoolean** (*name*, **args*, ***kwargs*)
Bases: *king_phisher.client.plugins.ClientOptionMixin*, *king_phisher.plugins.OptionBoolean*

__init__ (*name*, **args*, ***kwargs*)

Parameters

- **name** (*str*) – The name of this option.
- **description** (*str*) – The description of this option.

- **default** – The default value of this option.
- **display_name** (*str*) – The name to display in the UI to the user for this option.

class king_phisher.client.plugins.**ClientOptionEnum**(*name*, **args*, ***kwargs*)

Bases: *king_phisher.client.plugins.ClientOptionMixin*, *king_phisher.plugins.OptionEnum*

Parameters

- **name** (*str*) – The name of this option.
- **description** (*str*) – The description of this option.
- **choices** (*tuple*) – The supported values for this option.
- **default** – The default value of this option.
- **display_name** (*str*) – The name to display in the UI to the user for this option

__init__(*name*, **args*, ***kwargs*)

Parameters

- **name** (*str*) – The name of this option.
- **description** (*str*) – The description of this option.
- **default** – The default value of this option.
- **display_name** (*str*) – The name to display in the UI to the user for this option.

class king_phisher.client.plugins.**ClientOptionInteger**(*name*, **args*, ***kwargs*)

Bases: *king_phisher.client.plugins.ClientOptionMixin*, *king_phisher.plugins.OptionInteger*

__init__(*name*, **args*, ***kwargs*)

Parameters

- **name** (*str*) – The name of this option.
- **description** (*str*) – The description of this option.
- **default** – The default value of this option.
- **display_name** (*str*) – The name to display in the UI to the user for this option.
- **adjustment** (Gtk.Adjustment) – The adjustment details of the options value.

class king_phisher.client.plugins.**ClientOptionMixin**(*name*, **args*, ***kwargs*)

Bases: object

A mixin for options used by plugins for the client application. It provides additional methods for creating GTK widgets for the user to set the option's value as well as retrieve it.

__init__(*name*, **args*, ***kwargs*)

Parameters

- **name** (*str*) – The name of this option.
- **description** (*str*) – The description of this option.
- **default** – The default value of this option.
- **display_name** (*str*) – The name to display in the UI to the user for this option.

get_widget(*application*, *value*)
Create a widget suitable for configuring this option. This is meant to allow subclasses to specify and create an appropriate widget type.

Parameters

- **application** (Gtk.Application) – The parent application for this object.
- **value** – The initial value to set for this widget.

Returns The widget for the user to set the option with.

Return type Gtk.Widget

get_widget_value(*widget*)
Get the value of a widget previously created with *get_widget()*.

Parameters **widget** (Gtk.Widget) – The widget from which to retrieve the value from for this option.

Returns The value for this option as set in the widget.

class king_phisher.client.plugins.**ClientOptionPath**(*name*, **args*, ***kwargs*)
Bases: *king_phisher.client.plugins.ClientOptionString*

__init__(*name*, **args*, ***kwargs*)

Parameters

- **name** (*str*) – The name of this option.
- **description** (*str*) – The description of this option.
- **default** – The default value of this option.
- **display_name** (*str*) – The name to display in the UI to the user for this option.
- **path_type** (*str*) – The type of the path to select, either 'directory', 'file-open' or 'file-save'.

class king_phisher.client.plugins.**ClientOptionPort**(**args*, ***kwargs*)
Bases: *king_phisher.client.plugins.ClientOptionInteger*

__init__(**args*, ***kwargs*)

Parameters

- **name** (*str*) – The name of this option.
- **description** (*str*) – The description of this option.
- **default** – The default value of this option.
- **display_name** (*str*) – The name to display in the UI to the user for this option.

class king_phisher.client.plugins.**ClientOptionString**(*name*, **args*, ***kwargs*)
Bases: *king_phisher.client.plugins.ClientOptionMixin*, *king_phisher.plugins.OptionString*

__init__(*name*, **args*, ***kwargs*)

Parameters

- **name** (*str*) – The name of this option.
- **description** (*str*) – The description of this option.
- **default** – The default value of this option.

- **display_name** (*str*) – The name to display in the UI to the user for this option.

class king_phisher.client.plugins.**ClientPlugin**(*application*)
Bases: *king_phisher.plugins.PluginBase*

The base object to be inherited by plugins that are loaded into the King Phisher client. This provides a convenient interface for interacting with the runtime.

add_menu_item(*menu_path*, *handler=None*)
Add a new item into the main menu bar of the application. Menu items created through this method are automatically removed when the plugin is disabled. If no *handler* is specified, the menu item will be a separator, otherwise *handler* will automatically be connected to the menu item's `activate` signal.

Parameters

- **menu_path** (*str*) – The path to the menu item, delimited with > characters.
- **handler** – The optional callback function to be connected to the new `Gtk.MenuItem` instance's activate signal.

Returns The newly created and added menu item.

Return type `Gtk.MenuItem`

add_submenu(*menu_path*)
Add a submenu into the main menu bar of the application. Submenus created through this method are automatically removed when the plugin is disabled.

Parameters **menu_path** (*str*) – The path to the submenu, delimited with > characters.

Returns The newly created and added menu item.

Return type `Gtk.MenuItem`

application = None
A reference to the *KingPhisherClientApplication*.

config
A dictionary that can be used by this plugin for persistent storage of it's configuration.

signal_connect(*name*, *handler*, *gobject=None*, *after=False*)
Connect *handler* to a signal by *name* to an arbitrary GObject. Signals connected through this method are automatically cleaned up when the plugin is disabled. If no GObject is specified, the *application* instance is used.

> **Warning:** If the signal needs to be disconnected manually by the plugin, this method should not be used. Instead the handler id should be kept as returned by the GObject's native connect method.

Parameters

- **name** (*str*) – The name of the signal.
- **handler** (*function*) – The function to be invoked with the signal is emitted.
- **gobject** – The object to connect the signal to.
- **after** (*bool*) – Whether to call the user specified handler after the default signal handler or before.

signal_connect_server_event(*name*, *handler*, *event_types*, *attributes*)
Connect *handler* to the server signal with *name*. This method is similar to *signal_connect()* but also

sets up the necessary event subscriptions to ensure that the handler will be called. These event subscriptions are automatically cleaned up when the plugin is disabled.

> **Warning:** Server events are emitted based on the client applications event subscriptions. This means that while *handler* will be called for the event types specified, it may also be called for additional unspecified event types if other plugins have subscribed to them. This means that it is important to check the event type within the handler itself and react as necessary. To avoid this simply use the *event_type_filter()* decorator for the *handler* function.

Parameters

- **name** (*str*) – The name of the signal.
- **handler** – The function to be invoked with the signal is emitted.
- **event_types** (*list*) – A list of sub-types for the corresponding event.
- **attributes** (*list*) – A list of attributes of the event object to be sent to the client.

class king_phisher.client.plugins.**ClientPluginMailerAttachment**(**args*, ***kwargs*)

Bases: *king_phisher.client.plugins.ClientPlugin*

The base object to be inherited by plugins that intend to modify attachment files such as for inserting the tracking URL into them. Plugins which inherit from this base class must override the *process_attachment_file()* method which will automatically be called for each target a user is sending messages to.

process_attachment_file(*input_path*, *output_path*, *target*)

This function is automatically called for each target that a user is sending messages to. This method is intended to process the specified attachment file. This method removes the need to manually cleanup the *output_path* because it is handled automatically as necessary.

Parameters

- **input_path** (*str*) – The path to the input file to process. This path is guaranteed to be an existing file that is readable.
- **output_path** (*str*) – The path to optionally write the output file to. This path may or may not be the same as *input_path*. If the plugin needs to rename the file, to for example change the extension, then the new output_path must be returned.
- **target** (*MessageTarget*) – The target information for the messages intended recipient.

Returns None or an updated value for *output_path* in the case that the plugin renames it.

class king_phisher.client.plugins.**ClientPluginManager**(*path*, *application*)

Bases: *king_phisher.plugins.PluginManagerBase*

The manager for plugins loaded into the King Phisher client application.

client.server_events

Functions

king_phisher.client.server_events.**event_type_filter**(*event_types*, *is_method=False*)

A decorator to filter a signal handler by the specified event types. Using this will ensure that the decorated

function is only called for the specified event types and not others which may have been subscribed to elsewhere in the application.

Parameters

- **event_types** (*list, str*) – A single event type as a string or a list of event type strings.
- **is_method** (*bool*) – Whether or not the function being decorated is a class method.

Classes

class king_phisher.client.server_events.**ServerEventSubscriber**(*rpc*)
Bases: GObject.GObject

An object which provides functionality to subscribe to events that are published by the remote King Phisher server instance. This object manages the subscriptions and forwards the events allowing consumers to connect to the available GObject signals.

Note: Both the *ServerEventSubscriber.subscribe()* and *ServerEventSubscriber.unsubscribe()* methods of this object internally implement reference counting for the server events. This makes it possible for multiple subscriptions to be created and deleted without interfering with each other.

The socket is opened automatically when this object is initialized and will automatically attempt to reconnect if the connection is closed if the *reconnect* attribute is true. After initializing this object, check the *is_connected* attribute to ensure that it is properly connected to the server.

__init__(*rpc*)

Parameters **rpc** (*KingPhisherRPCClient*) – The client's connected RPC instance.

is_connected
True if the event socket is connected to the server.

is_subscribed(*event_id*, *event_type*)
Check if the client is currently subscribed to the specified server event.

Parameters

- **event_id** (*str*) – The identifier of the event to subscribe to.
- **event_type** (*str*) – A sub-type for the corresponding event.

Returns Whether or not the client is subscribed to the event.

Return type bool

reconnect = None
Whether or not the socket should attempt to reconnect itself when it has been closed.

shutdown()
Disconnect the event socket from the remote server. After the object is shutdown, remove events will no longer be published.

Parameters **timeout** (*int*) – An optional timeout for how long to wait on the worker thread.

subscribe(*event_id*, *event_types*, *attributes*)
Subscribe the client to the specified event published by the server. When the event is published the specified *attributes* of it and it's corresponding id and type information will be sent to the client.

Parameters

- **event_id** (*str*) – The identifier of the event to subscribe to.
- **event_types** (*list*) – A list of sub-types for the corresponding event.
- **attributes** (*list*) – A list of attributes of the event object to be sent to the client.

unsubscribe(*event_id*, *event_types*, *attributes*)
Unsubscribe from an event published by the server that the client previously subscribed to.

Parameters

- **event_id** (*str*) – The identifier of the event to subscribe to.
- **event_types** (*list*) – A list of sub-types for the corresponding event.
- **attributes** (*list*) – A list of attributes of the event object to be sent to the client.

client.web_cloner

This module contains the functionality used by the client to clone web pages.

Classes

class king_phisher.client.web_cloner.**ClonedResourceDetails**
A named tuple which contains details regard a resource that has been cloned.

resource
The web resource that has been cloned.

mime_type
The MIME type that was provided by the server for the cloned resource.

size
The size of the original resource that was provided by the server.

file_name
The path to the file which the resource was written to.

class king_phisher.client.web_cloner.**WebPageCloner**(*target_url*, *dest_dir*)
Bases: object

This object is used to clone web pages. It will use the WebKit2GTK+ engine and hook signals to detect what remote resources that are loaded from the target URL. These resources are then written to disk. Resources that have a MIME type of text/html have the King Phisher server javascript file patched in..

__init__(*target_url*, *dest_dir*)

Parameters

- **target_url** (*str*) – The URL of the target web page to clone.
- **dest_dir** (*str*) – The path of a directory to write the resources to.

cloned_resources = None
A collections.OrderedDict instance of *ClonedResourceDetails* keyed by the web resource they describe.

copy_resource_data(*resource*, *data*)
Copy the data from a loaded resource to a local file.

Parameters

- **resource** (WebKit2.WebResource) – The resource whose data is being copied.

- **data** (*bytes, str*) – The raw data of the represented resource.

patch_html(*data*, *encoding='utf-8'*)
Patch the HTML data to include the King Phisher javascript resource. The script tag is inserted just before the closing head tag. If no head tag is present, the data is left unmodified.

Parameters **data** (*str*) – The HTML data to patch.

Returns The patched HTML data.

Return type str

resource_is_on_target(*resource*)
Test whether the resource is on the target system. This tries to match the hostname, scheme and port number of the resource's URI against the target URI.

Returns Whether the resource is on the target or not.

Return type bool

stop_cloning()
Stop the current cloning operation if it is running.

wait()
Wait for the cloning operation to complete and return whether the operation was successful or not.

Returns True if the operation was successful.

Return type bool

server

This package contains all packages and modules specific to the server application.

server.database

database.manager

This module provides the functionality to manage the server application's database connection.

Functions

king_phisher.server.database.manager.**clear_database**()
Delete all data from all tables in the connected database. The database schema will remain unaffected.

> **Warning:** This action can not be reversed and there is no confirmation before it takes place.

king_phisher.server.database.manager.**export_database**(*target_file*)
Export the contents of the database using SQLAlchemy's serialization. This creates an archive file containing all of the tables and their data. The resulting export can be imported into another supported database so long as the *SCHEMA_VERSION* is the same.

Parameters **target_file** (*str*) – The file to write the export to.

king_phisher.server.database.manager.**import_database**(*target_file*, *clear=True*)
Import the contents of a serialized database from an archive previously created with the *export_database()* function. The current *SCHEMA_VERSION* must be the same as the exported archive.

> **Warning:** This will by default delete the contents of the current database in accordance with the *clear* parameter. If *clear* is not specified and objects in the database and import share an ID, they will be merged.

Parameters

- **target_file** (*str*) – The database archive file to import from.
- **clear** (*bool*) – Whether or not to delete the contents of the existing database before importing the new data.

king_phisher.server.database.manager.**normalize_connection_url**(*connection_url*)
Normalize a connection url by performing any conversions necessary for it to be used with the database API.

Parameters **connection_url** (*str*) – The connection url to normalize.

Returns The normalized connection url.

Return type str

king_phisher.server.database.manager.**get_meta_data**(*key*, *session=None*)
Retrieve the value from the database's metadata storage.

Parameters

- **key** (*str*) – The name of the value to retrieve.
- **session** – The session to use to retrieve the value.

Returns The meta data value.

king_phisher.server.database.manager.**get_row_by_id**(*session*, *table*, *row_id*)
Retrieve a database row from the specified table by it's unique id.

Parameters

- **session** (*.Session*) – The database session to use for the query.
- **table** – The table object or the name of the database table where the row resides.
- **row_id** – The id of the row to retrieve.

Returns The object representing the specified row or None if it does not exist.

king_phisher.server.database.manager.**init_alembic**(*engine*, *schema_version*)
Creates the alembic_version table and sets the value of the table according to the specified schema version.

Parameters

- **engine** (*sqlalchemy.engine.Engine*) – The engine used to connect to the database.
- **schema_version** (*int*) – The MetaData schema_version to set the alembic version to.

king_phisher.server.database.manager.**init_database**(*connection_url*, *extra_init=False*)
Create and initialize the database engine. This must be done before the session object can be used. This will also attempt to perform any updates to the database schema if the backend supports such operations.

Parameters

- **connection_url** (*str*) – The url for the database connection.
- **extra_init** (*bool*) – Run optional extra dbms-specific initialization logic.

Returns The initialized database engine.

king_phisher.server.database.manager.**init_database_postgresql**(*connection_url*)
Perform additional initialization checks and operations for a PostgreSQL database. If the database is hosted locally this will ensure that the service is currently running and start it if it is not. Additionally if the specified database or user do not exist, they will be created.

Parameters **connection_url** (sqlalchemy.engine.url.URL) – The url for the PostgreSQL database connection.

Returns The initialized database engine.

king_phisher.server.database.manager.**set_meta_data**(*key*, *value*, *session=None*)
Store a piece of metadata regarding the King Phisher database.

Parameters

- **key** (*str*) – The name of the data.
- **value** (*int, str*) – The value to store.
- **session** – The session to use to store the value.

database.models

This module provides the models for the data stored in the database as well as functionality for defining and managing the models themselves.

Data

king_phisher.server.database.models.**database_table_objects**
A dictionary which contains all the database tables and their primitive objects.

king_phisher.server.database.models.**database_tables**
A dictionary which contains all the database tables and their column names.

king_phisher.server.database.models.**SCHEMA_VERSION**
The schema version of the database, used for compatibility checks.

Functions

king_phisher.server.database.models.**current_timestamp**(**args*, ***kwargs*)
The function used for creating the timestamp used by database objects.

Returns The current timestamp.

Return type datetime.datetime

king_phisher.server.database.models.**get_tables_with_column_id**(*column_id*)
Get all tables which contain a column named *column_id*.

Parameters **column_id** (*str*) – The column name to get all the tables of.

Returns The list of matching tables.

Return type set

king_phisher.server.database.models.**register_table**(*table*)
Register a database table. This will populate the information provided in DATABASE_TABLES dictionary. This also forwards signals to the appropriate listeners within the server.signal module.

Parameters **table** (*cls*) – The table to register.

Classes

class king_phisher.server.database.models.**BaseRowCls**
Bases: object

The base class from which other database table objects inherit from. Provides a standard __repr__ method and default permission checks which are to be overridden as desired by subclasses.

assert_session_has_permissions(**args*, ***kwargs*)
A convenience function which wraps *session_has_permissions()* and raises a *KingPhisherPermissionError* if the session does not have the specified permissions.

is_private = False
Whether the table is only allowed to be accessed by the server or not.

session_has_permissions(*access*, *session*)
Check that the authenticated session has the permissions specified in *access*. The permissions in *access* are abbreviated with the first letter of create, read, update, and delete.

Parameters

- **access** (*str*) – The desired permissions.
- **session** – The authenticated session to check access for.

Returns Whether the session has the desired permissions.

Return type bool

database.storage

This module provides functionality to utilize the database for persistent storage.

Classes

class king_phisher.server.database.storage.**KeyValueStorage**(*namespace=None*)
This class provides key-value storage of arbitrary data in the database. The *serializers* module is used for converting data into a format suitable for storing in the database. This object, once initialized, provides an interface just like a standard dictionary object. An optional namespace can be specified as a unique identifier, allowing different sources to store data using the same keys. All keys must be strings but data can be anything that is serializable.

__init__(*namespace=None*)

Parameters **namespace** (*str*) – The unique identifier of this namespace.

serializer
alias of MsgPack

server.aaa

This module provides the functionality authentication authorization and access to the server application.

Functions

king_phisher.server.aaa.**get_groups_for_user**(*username*)
Get the groups that a user is a member of.

Parameters **username** (*str*) – The user to lookup group membership for.

Return type set

Returns The names of the groups that the user is a member of.

Classes

class king_phisher.server.aaa.**AuthenticatedSession**(*user*)
Bases: object

A container to store information associated with an authenticated session.

__init__(*user*)

Parameters **user** – The unique identifier for the authenticated user.

created

event_socket
An optional *EventSocket* associated with the client. If the client has not opened an event socket, this is None.

classmethod from_db_authenticated_session(*stored_session*)
Load an instance from a record stored in the database.

Parameters **stored_session** – The authenticated session from the database to load.

Returns A new *AuthenticatedSession* instance.

last_seen

user

class king_phisher.server.aaa.**AuthenticatedSessionManager**(*timeout='30m'*)
Bases: object

A container for managing authenticated sessions.

__init__(*timeout='30m'*)

Parameters **timeout** (*int, str*) – The length of time in seconds for which sessions are valid.

clean()
Remove sessions which have expired.

get(*session_id*, *update_timestamp=True*)
Look up an *AuthenticatedSession* instance from it's unique identifier and optionally update the last seen timestamp. If the session is not found or has expired, None will be returned.

Parameters

- **session_id** (*str*) – The unique identifier of the session to retrieve.

- **update_timestamp** (*bool*) – Whether or not to update the last seen timestamp for the session.

Returns The session if it exists and is active.

Return type *AuthenticatedSession*

put(*user*)
Create and store a new *AuthenticatedSession* object for the specified user id. Any previously existing sessions for the specified user are removed from the manager.

Returns The unique identifier for this session.

Return type str

remove(*session_id*)
Remove the specified session from the manager.

Parameters **session_id** (*str*) – The unique identifier for the session to remove.

stop()

class king_phisher.server.aaa.**CachedPassword**(*pw_hash*)
Bases: object

A cached in-memory password. Cleartext passwords are salted with data generated at runtime and hashed before being stored for future comparisons.

__init__(*pw_hash*)

Parameters **pw_hash** (*bytes*) – The salted hash of the password to cache.

hash_algorithm = 'sha512'

iterations = 5000

classmethod new_from_password(*password*)
Create a new instance from a plaintext password.

Parameters **password** (*str*) – The password to cache in memory.

pw_hash

salt = 'om-N!x,-&R'

time

class king_phisher.server.aaa.**ForkedAuthenticator**(*cache_timeout='10m'*, *required_group=None*, *pam_service='sshd'*)
Bases: object

This provides authentication services to the King Phisher server through PAM. It is initialized while the server is running as root and forks into the background before the privileges are dropped. The child continues to run as root and forwards requests to PAM on behalf of the parent process which is then free to drop privileges. The pipes use JSON to encode the request data as a string before sending it and using a newline character as the terminator. Requests from the parent process to the child process include a sequence number which must be included in the response.

__init__(*cache_timeout='10m'*, *required_group=None*, *pam_service='sshd'*)

Parameters

- **cache_timeout** (*int, str*) – The life time of cached credentials in seconds.
- **required_group** (*str*) – A group that if specified, users must be a member of to be authenticated.

- **pam_service** (*str*) – The service to use for identification to pam when authenticating.

authenticate(*username*, *password*)
Check if a username and password are valid. If they are, the password will be salted, hashed with SHA-512 and stored so the next call with the same values will not require sending a request to the forked child.

Parameters

- **username** (*str*) – The username to check.
- **password** (*str*) – The password to check.

Returns Whether the credentials are valid or not.

Return type bool

cache = None
The credential cache dictionary. Keys are usernames and values are tuples of password hashes and ages.

cache_timeout = None
The timeout of the credential cache in seconds.

child_pid = None
The PID of the forked child.

child_routine()
The main routine that is executed by the child after the object forks. This loop does not exit unless a stop request is made.

response_timeout = None
The timeout for individual requests in seconds.

send(*request*)
Encode and send a request through the pipe to the opposite end. This also sets the 'sequence' member of the request and increments the stored value.

Parameters **request** (*dict*) – A request.

sequence_number = None
A sequence number to use to align requests with responses.

stop()
Send a stop request to the child process and wait for it to exit.

server.build

This module contains the functionality to build a new server instance from a configuration file. This intends to keep the error checking logic for potential configuration problems contained.

Functions

king_phisher.server.build.**get_bind_addresses**(*config*)
Retrieve the addresses on which the server should bind to. Each of these addresses should be an IP address, port and optionally enable SSL. The returned list will contain tuples for each address found in the configuration. These tuples will be in the (host, port, use_ssl) format that is compatible with AdvancedHTTPServer.

Parameters **config** (smoke_zephyr.configuration.Configuration) – Configuration to retrieve settings from.

Returns The specified addresses to bind to.

Return type list

king_phisher.server.build.**get_ssl_hostnames**(*config*)

Retrieve the SSL hosts that are specified within the configuration. This also ensures that the settings appear to be valid by ensuring that the necessary files are defined and readable.

Parameters **config** (smoke_zephyr.configuration.Configuration) – Configuration to retrieve settings from.

Returns The specified SSH hosts.

Return type list

king_phisher.server.build.**server_from_config**(*config*, *handler_klass=None*, *plugin_manager=None*)

Build a server from a provided configuration instance. If *handler_klass* is specified, then the object must inherit from the corresponding KingPhisherServer base class.

Parameters

- **config** (smoke_zephyr.configuration.Configuration) – Configuration to retrieve settings from.
- **handler_klass** (*KingPhisherRequestHandler*) – Alternative handler class to use.
- **plugin_manager** (*ServerPluginManager*) – The server's plugin manager instance.

Returns A configured server instance.

Return type *KingPhisherServer*

server.graphql

This module provides the GraphQL interface for querying information from the King Phisher server. This allows flexibility in how the client would like for the returned data to be formatted. This interface can be accessed directly by the server or through the RPC end point at *rpc_graphql()*.

Classes

class king_phisher.server.graphql.**AuthorizationMiddleware**

Bases: object

An authorization provider to ensure that the permissions on the objects that are queried are respected. If no **rpc_session** key is provided in the **context** dictionary then no authorization checks can be performed and all objects and operations will be accessible. The **rpc_session** key's value must be an instance of *AuthenticatedSession*.

class king_phisher.server.graphql.**Query**(**args*, ***kwargs*)

Bases: graphene.types.objecttype.ObjectType

This is the root query object used for GraphQL queries.

class king_phisher.server.graphql.**Schema**(***kwargs*)

Bases: graphene.types.schema.Schema

This is the top level schema object for GraphQL. It automatically sets up sane defaults to be used by the King Phisher server including setting the query to *Query* and adding the *AuthorizationMiddleware* to each execution.

server.pages

This module provides functionality for Jinja functions used to generate server page content.

Functions

king_phisher.server.pages.**embed_youtube_video**(*video_id*, *autoplay=True*, *enable_js=False*, *start=0*, *end=None*)

A Jinja function to embed a video into a web page using YouTube's iframe API. In order to enable a training button after the video has ended the youtube.js file needs to be included and *enable_js* just be set to True. If *start* or *end* are specified as strings, the must be in a format suitable to be parsed by `parse_timespan()`.

Parameters

- **video_id** (*str*) – The id of the YouTube video to embed.
- **autoplay** (*bool*) – Start playing the video as soon as the page loads.
- **enable_js** (*bool*) – Enable the Javascript API.
- **start** (*int, str*) – The time offset at which the video should begin playing.
- **end** (*int, str*) – The time offset at which the video should stop playing.

king_phisher.server.pages.**make_csrf_page**(*url*, *params*, *method='POST'*)

A Jinja function which will create an HTML page that will automatically perform a CSRF attack against another page.

Parameters

- **url** (*str*) – The URL to use as the form action.
- **params** (*dict*) – The parameters to send in the forged request.
- **method** (*str*) – The HTTP method to use when submitting the form.

king_phisher.server.pages.**make_redirect_page**(*url*, *title='Automatic Redirect'*)

A Jinja function which will create an HTML page that will automatically redirect the viewer to a different url.

Parameters

- **url** (*str*) – The URL to redirect the user to.
- **title** (*str*) – The title to use in the resulting HTML page.

server.plugins

Classes

class king_phisher.server.plugins.**ServerPlugin**(*root_config*)

Bases: *king_phisher.plugins.PluginBase*

The base object to be inherited by plugins that are loaded into the King Phisher server. This provides a convenient interface for interacting with the runtime.

config

A dictionary that can be used by this plugin to access it's configuration. Any changes to this configuration will be lost with the server restarts.

register_http(*path*, *method*)
Register a new HTTP request handler at *path* that is handled by *method*. Two parameters are passed to the method. The first parameter is a *KingPhisherRequestHandler* instance and the second is a dictionary of the HTTP query parameters. The specified path is added within the plugins private HTTP handler namespace at `_/plugins/$PLUGIN_NAME/$PATH`

> **Warning:** This resource can be reached by any user whether or not they are authenticated and or associated with a campaign.

Parameters

- **path** (*str*) – The path to register the method at.
- **method** – The handler for the HTTP method.

register_rpc(*path*, *method*)
Register a new RPC function at *path* that is handled by *method*. This RPC function can only be called by authenticated users. A single parameter of the *KingPhisherRequestHandler* instance is passed to *method* when the RPC function is invoked. The specified path is added within the plugins private RPC handler namespace at `plugins/$PLUGIN_NAME/$PATH`.

Parameters

- **path** (*str*) – The path to register the method at.
- **method** – The handler for the RPC method.

root_config = None
A reference to the main server instance *config*.

server = None
A reference to the *KingPhisherServer* instance. Only available if the instance has been created.

storage = None
An instance of *KeyValueStorage* for this plugin to use for persistent data storage. This attribute is None until the *db_initialized* signal is emitted.

class king_phisher.server.plugins.**ServerPluginManager**(*config*)
Bases: *king_phisher.plugins.PluginManagerBase*

The manager for plugins loaded into the King Phisher server application.

server.rest_api

This module provides the functionality exposed by the server application's REST API.

Data

king_phisher.server.rest_api.**REST_API_BASE**
The base URI path for REST API requests.

Functions

king_phisher.server.rest_api.**generate_token**()
Generate the token to be checked when REST API requests are made.

Returns The API token

Return type str

king_phisher.server.rest_api.**rest_handler**(*handle_function*)
A function for decorating REST API handlers. The function checks the API token in the request and encodes the handler response in JSON to be sent to the client.

Parameters **handle_function** – The REST API handler.

server.server

This module contains the functionality that provides the application's low-level HTTP server logic.

Classes

class king_phisher.server.server.**KingPhisherRequestHandler**(**args*, ***kwargs*)
Bases: advancedhttpserver.RequestHandler

adjust_path()
Adjust the *path* attribute based on multiple factors.

campaign_id
The campaign id that is associated with the current request's visitor. This is retrieved by looking up the *message_id* value in the database. If no campaign is associated, this value is None.

config = None
A reference to the main server instance *KingPhisherServer.config*.

get_client_ip()
Intelligently get the IP address of the HTTP client, optionally accounting for proxies that may be in use.

Returns The clients IP address.

Return type str

get_query_creds(*check_query=True*)
Get credentials that have been submitted in the request. For credentials to be returned at least a username must have been specified. The returned username will be None or a non-empty string. The returned password will be None if the parameter was not found or a string which maybe empty. This functions checks the query data for credentials first if *check_query* is True, and then checks the contents of an Authorization header.

Parameters **check_query** (*bool*) – Whether or not to check the query data in addition to an Authorization header.

Returns The submitted credentials.

Return type tuple

get_template_vars_client()
Build a dictionary of variables for a client with an associated campaign.

Returns The client specific template variables.

Return type dict

issue_alert(*alert_text*, *campaign_id*)
Send an SMS alert. If no *campaign_id* is specified all users with registered SMS information will receive the alert otherwise only users subscribed to the campaign specified.

Parameters

- **alert_text** (*str*) – The message to send to subscribers.
- **campaign_id** (*int*) – The campaign subscribers to send the alert to.

message_id
The message id that is associated with the current request's visitor. This is retrieved by looking at an 'id' parameter in the query and then by checking the *visit_id* value in the database. If no message id is associated, this value is None. The resulting value will be either a confirmed valid value, or the value of the configurations server.secret_id for testing purposes.

path = None
The resource path of the current HTTP request.

vhost
The value of the Host HTTP header.

visit_id
The visit id that is associated with the current request's visitor. This is retrieved by looking for the King Phisher cookie. If no cookie is set, this value is None.

class king_phisher.server.server.**KingPhisherServer**(*config*, *plugin_manager*, *handler_klass*, **args*, ***kwargs*)
Bases: advancedhttpserver.AdvancedHTTPServer

The main HTTP and RPC server for King Phisher.

__init__(*config*, *plugin_manager*, *handler_klass*, **args*, ***kwargs*)

Parameters **config** (smoke_zephyr.configuration.Configuration) – Configuration to retrieve settings from.

config = None
A Configuration instance used as the main King Phisher server configuration.

headers = None
A OrderedDict containing additional headers specified from the server configuration to include in responses.

job_manager = None
A JobManager instance for scheduling tasks.

shutdown(**args*, ***kwargs*)
Request that the server perform any cleanup necessary and then shut down. This will wait for the server to stop before it returns.

server.server_rpc

This module provides the RPC server functionality that is used by the client to communicate with the server application.

Data

king_phisher.server.server_rpc.**CONFIG_READABLE**
Configuration options that can be accessed by the client.

king_phisher.server.server_rpc.**CONFIG_WRITEABLE**
Configuration options that can be changed by the client at run time.

king_phisher.server.server_rpc.**RPC_AUTH_HEADER = 'X-RPC-Auth'**
The header which contains the RPC authorization / session token.

king_phisher.server.server_rpc.**VIEW_ROW_COUNT = 50**
The default number of rows to return when one of the /view methods are called.

Functions

king_phisher.server.server_rpc.**register_rpc**(*path*, *database_access=False*, *log_call=False*)
Register an RPC function with the HTTP request handler. This allows the method to be remotely invoked using King Phisher's standard RPC interface. If *database_access* is specified, a SQLAlchemy session will be passed as the second argument, after the standard `RequestHandler` instance.

Parameters

- **path** (*str*) – The path for the RPC function.
- **database_access** (*bool*) – Whether or not the function requires database access.
- **log_call** (*bool*) – Whether or not to log the arguments which the function is called with.

king_phisher.server.server_rpc.**rpc_campaign_alerts_is_subscribed**(*handler_instance*, **args*, ***kwargs*)
Check if the user is subscribed to alerts for the specified campaign.

Parameters **campaign_id** (*int*) – The ID of the campaign.

Returns The alert subscription status.

Return type bool

king_phisher.server.server_rpc.**rpc_campaign_alerts_subscribe**(*handler_instance*, **args*, ***kwargs*)
Subscribe to alerts for the specified campaign.

Parameters **campaign_id** (*int*) – The ID of the campaign.

king_phisher.server.server_rpc.**rpc_campaign_alerts_unsubscribe**(*handler_instance*, **args*, ***kwargs*)
Unsubscribe to alerts for the specified campaign.

Parameters **campaign_id** (*int*) – The ID of the campaign.

king_phisher.server.server_rpc.**rpc_campaign_landing_page_new**(*handler_instance*, **args*, ***kwargs*)
Add a landing page for the specified campaign. Landing pages refer to resources that when visited by a user should cause the visit counter to be incremented.

Parameters

- **campaign_id** (*int*) – The ID of the campaign.
- **hostname** (*str*) – The hostname which will be used to serve the request.
- **page** (*str*) – The request resource.

king_phisher.server.server_rpc.**rpc_campaign_message_new**(*handler_instance*, **args*, ***kwargs*)
Record a message that has been sent as part of a campaign. These details can be retrieved later for value substitution in template pages.

Parameters

- **campaign_id** (*int*) – The ID of the campaign.
- **email_id** (*str*) – The message id of the sent email.
- **target_email** (*str*) – The email address that the message was sent to.
- **first_name** (*str*) – The first name of the message's recipient.
- **last_name** (*str*) – The last name of the message's recipient.
- **department_name** (*str*) – The name of the company department that the message's recipient belongs to.

king_phisher.server.server_rpc.**rpc_campaign_new**(*handler_instance*, **args*, ***kwargs*)
Create a new King Phisher campaign and initialize the database information.

Parameters

- **name** (*str*) – The new campaign's name.
- **description** (*str*) – The new campaign's description.

Returns The ID of the new campaign.

Return type int

king_phisher.server.server_rpc.**rpc_campaign_stats**(*handler_instance*, **args*, ***kwargs*)
Generate statistics regarding the specified campaign and return them in a dictionary. The dictionary will contain the keys credentials, credentials-unique, messages, messages-trained, visits, visits-unique. Values with unique in the key are counted unique by the message id for which they are associated.

Parameters **campaign_id** – The unique ID of the campaign to generate statistics for.

Returns The statistics for the specified campaign.

Return type dict

king_phisher.server.server_rpc.**rpc_config_get**(*handler_instance*, **args*, ***kwargs*)
Retrieve a value from the server's configuration.

Parameters **option_name** (*str*) – The name of the configuration option.

Returns The option's value.

king_phisher.server.server_rpc.**rpc_config_set**(*handler_instance*, **args*, ***kwargs*)
Set options in the server's configuration. Any changes to the server's configuration are not written to disk.

Parameters **options** (*dict*) – A dictionary of option names and values

king_phisher.server.server_rpc.**rpc_events_is_subscribed**(*handler_instance*, **args*, ***kwargs*)
Check if the client is currently subscribed to the specified server event.

Parameters

- **event_id** (*str*) – The identifier of the event to subscribe to.
- **event_type** (*str*) – A sub-type for the corresponding event.

Returns Whether or not the client is subscribed to the event.

Return type bool

king_phisher.server.server_rpc.**rpc_events_subscribe**(*handler_instance*, **args*, ***kwargs*)

Subscribe the client to the specified event published by the server. When the event is published the specified *attributes* of it and it's corresponding id and type information will be sent to the client.

Parameters

- **event_id** (*str*) – The identifier of the event to subscribe to.
- **event_types** (*list*) – A list of sub-types for the corresponding event.
- **attributes** (*list*) – A list of attributes of the event object to be sent to the client.

king_phisher.server.server_rpc.**rpc_events_unsubscribe**(*handler_instance*, **args*, ***kwargs*)

Unsubscribe from an event published by the server that the client previously subscribed to.

Parameters

- **event_id** (*str*) – The identifier of the event to subscribe to.
- **event_types** (*list*) – A list of sub-types for the corresponding event.
- **attributes** (*list*) – A list of attributes of the event object to be sent to the client.

king_phisher.server.server_rpc.**rpc_database_count_rows**(*handler_instance*, **args*, ***kwargs*)

Get a count of the rows in the specified table where the search criteria matches.

Parameters

- **table_name** (*str*) – The name of the database table to query.
- **query_filter** (*dict*) – A dictionary mapping optional search criteria for matching the query.

Returns The number of matching rows.

Return type int

king_phisher.server.server_rpc.**rpc_database_delete_row_by_id**(*handler_instance*, **args*, ***kwargs*)

Delete the row from the table with the specified value in the id column. If the row does not exist, no error is raised.

Parameters

- **table_name** (*str*) – The name of the database table to delete a row from.
- **row_id** – The id value.

king_phisher.server.server_rpc.**rpc_database_delete_rows_by_id**(*handler_instance*, **args*, ***kwargs*)

Delete multiple rows from a table with the specified values in the id column. If a row id specified in *row_ids* does not exist, then it will be skipped and no error will be thrown.

Parameters

- **table_name** (*str*) – The name of the database table to delete rows from.
- **row_ids** (*list*) – The row ids to delete.

Returns The row ids that were deleted.

Return type list

king_phisher.server.server_rpc.**rpc_database_get_row_by_id**(*handler_instance*, **args*, ***kwargs*)

Retrieve a row from a given table with the specified value in the id column.

Parameters

- **table_name** (*str*) – The name of the database table to retrieve a row from.
- **row_id** – The id value.

Returns The specified row data.

Return type dict

king_phisher.server.server_rpc.**rpc_database_insert_row**(*handler_instance*, **args*, ***kwargs*)

Insert a new row into the specified table.

Parameters

- **table_name** (*str*) – The name of the database table to insert a new row into.
- **keys** (*list*) – The column names of *values*.
- **values** (*list*) – The values to be inserted in the row.

Returns The id of the new row that has been added.

king_phisher.server.server_rpc.**rpc_database_set_row_value**(*handler_instance*, **args*, ***kwargs*)

Set values for a row in the specified table with an id of *row_id*.

Parameters

- **table_name** (*str*) – The name of the database table to set the values of the specified row.
- **keys** (*tuple*) – The column names of *values*.
- **values** (*tuple*) – The values to be updated in the row.

king_phisher.server.server_rpc.**rpc_database_view_rows**(*handler_instance*, **args*, ***kwargs*)

Retrieve the rows from the specified table where the search criteria matches.

Parameters

- **table_name** (*str*) – The name of the database table to query.
- **page** (*int*) – The page number to retrieve results for.
- **query_filter** (*dict*) – A dictionary mapping optional search criteria for matching the query.

Returns A dictionary with columns and rows keys.

Return type dict

king_phisher.server.server_rpc.**rpc_geoip_lookup**(*handler_instance*, **args*, ***kwargs*)

Look up an IP address in the servers GeoIP database. If the IP address can not be found in the database, None will be returned.

Parameters

- **ip** (*str*) – The IP address to look up.
- **lang** (*str*) – The language to prefer for regional names.

Returns The geographic information for the specified IP address.

Return type dict

king_phisher.server.server_rpc.**rpc_geoip_lookup_multi**(*handler_instance*, **args*, ***kwargs*)

Look up multiple IP addresses in the servers GeoIP database. Each IP address that can not be found in the database will have its result set to None.

Parameters

- **ips** (*list*) – The list of IP addresses to look up.
- **lang** (*str*) – The language to prefer for regional names.

Returns A dictionary containing the results keyed by the specified IP addresses.

Return type dict

king_phisher.server.server_rpc.**rpc_graphql**(*handler_instance*, **args*, ***kwargs*)

Execute a GraphQL query and return the results. If the query fails to execute the errors returned are populated in the **errors** key of the results dictionary. If the query executes successfully the returned data is available in the **data** key of the results dictionary.

Parameters

- **query** (*str*) – The GraphQL query to execute.
- **query_vars** (*dict*) – Any variables needed by the *query*.

Returns The results of the query as a dictionary.

Return type dict

king_phisher.server.server_rpc.**rpc_login**(*handler_instance*, **args*, ***kwargs*)

king_phisher.server.server_rpc.**rpc_logout**(*handler_instance*, **args*, ***kwargs*)

king_phisher.server.server_rpc.**rpc_ping**(*handler_instance*, **args*, ***kwargs*)

An RPC method that can be used by clients to assert the status and responsiveness of this server.

Returns This method always returns True.

Return type bool

king_phisher.server.server_rpc.**rpc_plugins_list**(*handler_instance*, **args*, ***kwargs*)

Return information regarding enabled plugins in the server.

Returns A dictionary representing enabled plugins and their meta-data.

Return type dict

king_phisher.server.server_rpc.**rpc_shutdown**(*handler_instance*, **args*, ***kwargs*)

This method can be used to shut down the server. This function will return, however no subsequent requests will be processed.

> **Warning:** This action will stop the server process and there is no confirmation before it takes place.

king_phisher.server.server_rpc.**rpc_version**(*handler_instance*, **args*, ***kwargs*)

Get the version information of the server. This returns a dictionary with keys of version, version_info and rpc_api_version. These values are provided for the client to determine compatibility.

Returns A dictionary with version information.

Return type dict

server.signals

This module contains the signals which are used by the server to dispatch events. Additional signal details regarding how these signals are used is available in the *Server Signals* documentation.

Functions

king_phisher.server.signals.**safe_send**(*signal*, *logger*, **args*, ***kwargs*)

Send a signal and catch any exception which may be raised during it's emission. Details regarding the error that occurs (including a stack trace) are logged to the specified *logger*. This is suitable for allowing signals to be emitted in critical code paths without interrupting the emitter.

Parameters

- **signal** (*str*) – The name of the signal to send safely.
- **logger** (logging.Logger) – The logger to use for logging exceptions.
- **args** – The arguments to be forwarded to the signal as it is sent.
- **kwargs** – The key word arguments to be forward to the signal as it is sent.

Signals

king_phisher.server.signals.**credentials_received**

Sent when a new pair of credentials have been submitted.

Parameters

- **request_handler** – The handler for the received request.
- **username** (*str*) – The username of the credentials that were submitted.
- **password** (*str*) – The password of the credentials that were submitted.

king_phisher.server.signals.**db_initialized**

Emitted after a connection has been made and the database has been fully initialized. At this point, it is safe to operate on the database.

Parameters **connection_url** (sqlalchemy.engine.url.URL) – The connection string for the database that has been initialized.

king_phisher.server.signals.**db_session_deleted**

Emitted after one or more rows have been deleted on a SQLAlchemy session. At this point, references are valid but objects can not be modified. See sqlalchemy.orm.events.SessionEvents.after_flush() for more details.

Parameters

- **table** (*str*) – The name of the table for which the target objects belong.
- **targets** (*tuple*) – The objects that have been deleted with the session.
- **session** (sqlalchemy.orm.session.Session) – The SQLAlchemy session with which the *targets* are associated.

king_phisher.server.signals.**db_session_inserted**

Emitted after one or more rows have been inserted in a SQLAlchemy session. At this point, references are valid but objects can not be modified. See sqlalchemy.orm.events.SessionEvents.after_flush() for more details.

Parameters

- **table** (*str*) – The name of the table for which the target objects belong.
- **targets** (*tuple*) – The objects that have been inserted with the session.
- **session** (sqlalchemy.orm.session.Session) – The SQLAlchemy session with which the *targets* are associated.

king_phisher.server.signals.**db_session_updated**

Emitted after one or more rows have been updated in a SQLAlchemy session. At this point, references are valid but objects can not be modified. See sqlalchemy.orm.events.SessionEvents.after_flush() for more details.

Parameters

- **table** (*str*) – The name of the table for which the target objects belong.
- **targets** (*tuple*) – The objects that have been updated with the session.
- **session** (sqlalchemy.orm.session.Session) – The SQLAlchemy session with which the *targets* are associated.

king_phisher.server.signals.**db_table_delete**

Emitted before a row inheriting from Base is deleted from the database table. To only subscribe to delete events for a specific table, specify the table's name as the *sender* parameter when calling blinker.base.Signal.connect(). See sqlalchemy.orm.events.MapperEvents.before_delete() for more details.

Parameters

- **table** (*str*) – The name of the table for which the target object belongs.
- **mapper** (sqlalchemy.orm.mapper.Mapper) – The Mapper object which is the target of the event.
- **connection** (sqlalchemy.engine.Connection) – The SQLAlchemy connection object which is being used to emit the SQL statements for the instance.
- **target** – The target object instance.

king_phisher.server.signals.**db_table_insert**

Emitted before a row inheriting from Base is inserted into the database table. To only subscribe to insert events for a specific table, specify the table's name as the *sender* parameter when calling blinker.base.Signal.connect(). See sqlalchemy.orm.events.MapperEvents.before_insert() for more details.

Parameters

- **table** (*str*) – The name of the table for which the target object belongs.
- **mapper** (sqlalchemy.orm.mapper.Mapper) – The Mapper object which is the target of the event.
- **connection** (sqlalchemy.engine.Connection) – The SQLAlchemy connection object which is being used to emit the SQL statements for the instance.
- **target** – The target object instance.

king_phisher.server.signals.**db_table_update**

Emitted before a row inheriting from Base is updated in the database table. To only subscribe to update events for a specific table, specify the table's name as the *sender* parameter when calling blinker.base.Signal.connect(). See sqlalchemy.orm.events.MapperEvents.before_update() for more details.

Parameters

- **table** (*str*) – The name of the table for which the target object belongs.

- **mapper** (sqlalchemy.orm.mapper.Mapper) – The Mapper object which is the target of the event.
- **connection** (sqlalchemy.engine.Connection) – The SQLAlchemy connection object which is being used to emit the SQL statements for the instance.
- **target** – The target object instance.

king_phisher.server.signals.**email_opened**
Sent when a request for the embedded image is received.

Parameters **request_handler** – The handler for the received request.

king_phisher.server.signals.**request_received**
Sent when a new HTTP request has been received and is about to be processed.

Parameters **request_handler** – The handler for the received request.

king_phisher.server.signals.**response_sent**
Sent after a response to an HTTP request has been sent to the client. At this point headers may be added to the response body.

Parameters

- **request_handler** – The handler for the received request.
- **code** (*int*) – The HTTP status code that was sent in the response.
- **message** (*str*) – The HTTP message that was sent in the response.

king_phisher.server.signals.**rpc_method_call**
Sent when a new RPC request has been received and it's corresponding method is about to be called.

Parameters

- **method** (*str*) – The RPC method which is about to be executed.
- **request_handler** – The handler for the received request.
- **args** (*tuple*) – The arguments that are to be passed to the method.
- **kwargs** (*dict*) – The key word arguments that are to be passed to the method.

king_phisher.server.signals.**rpc_method_called**
Sent after an RPC request has been received and it's corresponding method has been called.

Parameters

- **method** (*str*) – The RPC method which has been executed.
- **request_handler** – The handler for the received request.
- **args** (*tuple*) – The arguments that were passed to the method.
- **kwargs** (*dict*) – The key word arguments that were passed to the method.
- **retval** – The value returned from the RPC method invocation.

king_phisher.server.signals.**rpc_user_logged_in**
Sent when a new RPC user has successfully logged in and created a new authenticated session.

Parameters

- **request_handler** – The handler for the received request.
- **session** (*str*) – The session ID of the newly logged in user.
- **name** (*str*) – The username of the newly logged in user.

king_phisher.server.signals.**rpc_user_logged_out**
Sent when an authenticated RPC user has successfully logged out and terminated their authenticated session.

Parameters

- **request_handler** – The handler for the received request.
- **session** (*str*) – The session ID of the user who has logged out.
- **name** (*str*) – The username of the user who has logged out.

king_phisher.server.signals.**server_initialized**
Sent when a new instance of *KingPhisherServer* is initialized.

Parameters **server** – The newly initialized server instance.

king_phisher.server.signals.**visit_received**
Sent when a new visit is received on a landing page. This is only emitted when a new visit entry is added to the database.

Parameters **request_handler** – The handler for the received request.

server.web_sockets

Classes

class king_phisher.server.web_sockets.**Event**(*event_id*, *event_type*, *sources*)
Bases: object

An object representing an event which occurred on the server in a way that is ready to be published to client subscribers.

event_id
The unique string identifier of this event.

event_type
The unique string identifier of the type of this event.

sources
The source objects which are associated with this event.

class king_phisher.server.web_sockets.**EventSocket**(*handler*, *manager*)
Bases: advancedhttpserver.WebSocketHandler

A socket through which server events are published to subscribers. This socket will automatically add and remove itself from the manager that is initialized with.

__init__(*handler*, *manager*)

Parameters

- **handler** (advancedhttpserver.RequestHandler) – The request handler that should be used by this socket.
- **manager** (*WebSocketsManager*) – The manager that this event socket should register with.

is_subscribed(*event_id*, *event_type*)
Check if the client is currently subscribed to the specified server event.

Parameters

- **event_id** (*str*) – The identifier of the event to subscribe to.

- **event_type** (*str*) – A sub-type for the corresponding event.

Returns Whether or not the client is subscribed to the event.

Return type bool

publish(*event*)
Publish the event by sending the relevant information to the client. If the client has not requested to receive the information through a subscription, then no data will be sent.

Parameters **event** (*Event*) – The object representing the data to be published.

subscribe(*event_id*, *event_types=None*, *attributes=None*)
Subscribe the client to the specified event published by the server. When the event is published the specified *attributes* of it and it's corresponding id and type information will be sent to the client.

Parameters

- **event_id** (*str*) – The identifier of the event to subscribe to.
- **event_types** (*list*) – A list of sub-types for the corresponding event.
- **attributes** (*list*) – A list of attributes of the event object to be sent to the client.

unsubscribe(*event_id*, *event_types=None*, *attributes=None*)
Unsubscribe from an event published by the server that the client previously subscribed to.

Parameters

- **event_id** (*str*) – The identifier of the event to subscribe to.
- **event_types** (*list*) – A list of sub-types for the corresponding event.
- **attributes** (*list*) – A list of attributes of the event object to be sent to the client.

class king_phisher.server.web_sockets.**WebSocketsManager**(*config*, *job_manager*)
Bases: object

An object used to manage connected web sockets.

__init__(*config*, *job_manager*)

Parameters

- **config** (smoke_zephyr.configuration.Configuration) – Configuration to retrieve settings from.
- **job_manager** (smoke_zephyr.job.JobManager) – A job manager instance that can be used to schedule tasks.

add(*web_socket*)
Add a connected web socket to the manager.

Parameters **web_socket** (advancedhttpserver.WebSocketHandler) – The connected web socket.

dispatch(*handler*)
A method that is suitable for use as a web_socket_handler.

Parameters **handler** (*KingPhisherRequestHandler*) – The current request handler instance.

logger = <logging.Logger object>

ping_all()
Ping all of the connected web sockets to ensure they stay alive. This method is automatically executed periodically through a job added when the manager is initialized.

remove (*web_socket*)
Remove a connected web socket from those that are currently being managed. If the web socket is not currently being managed, no changes are made.

Parameters **web_socket** (advancedhttpserver.WebSocketHandler) – The connected web socket.

stop ()
Shutdown the manager and clean up the resources it has allocated.

archive

This module provides a generic means to combine data and files into a single archive file.

Functions

king_phisher.archive.**is_archive** (*file_path*)
Check if the specified file appears to be a valid archive file that can be opened with *ArchiveFile*.

Parameters **file_path** (*str*) – The path to the file to check.

Returns Whether or not the file looks like a compatible archive.

Return type bool

Classes

class king_phisher.archive.**ArchiveFile** (*file_name*, *mode*, *encoding='utf-8'*)
An object representing a generic archive for storing information. The resulting archive file is a tarfile that can easily be opened and manipulated with external tools. This class also facilitates storing metadata with the archive. This metadata contains basic information such as the version of King Phisher that generated it, and a UTC timestamp of when it was created.

__init__ (*file_name*, *mode*, *encoding='utf-8'*)

Parameters

- **file_name** (*str*) – The path to the file to open as an archive.
- **mode** (*str*) – The mode to open the file such as 'r' or 'w'.
- **encoding** (*str*) – The encoding to use for strings.

add_data (*name*, *data*)
Add arbitrary data directly to the archive under the specified name. This allows data to be directly inserted into the archive without first writing it to a file or file like object.

Parameters

- **name** (*str*) – The name of the destination file in the archive.
- **data** (*bytes, str*) – The data to place into the archive.

add_file (*name*, *file_path*, *recursive=True*)
Place a file or directory into the archive. If *file_path* is a directory, it's contents will be added recursively if *recursive* is True.

Parameters

- **name** (*str*) – The name of the destination file in the archive.
- **file_path** (*str*) – The path to the file to add to the archive.
- **recursive** (*bool*) – Whether or not to add directory contents.

close()
Close the handle to the archive.

file_names
This property is a generator which yields the names of all of the contained files. The metadata file is skipped.

Returns A generator which yields all the contained file names.

Return type str

files
This property is a generator which yields tuples of two objects each where the first is the file name and the second is the file object. The metadata file is skipped.

Returns A generator which yields all the contained file name and file objects.

Return type tuple

get_data(*name*)
Return the data contained within the specified archive file.

Parameters **name** (*str*) – The name of the source file in the archive.

Returns The contents of the specified file.

Return type bytes

get_file(*name*)
Return the specified file object from the archive.

Parameters **name** (*str*) – The name of the source file in the archive.

Returns The specified file.

Return type file

has_file(*name*)
Check if a file exists within archive.

Parameters **name** (*str*) –

Returns Whether or not the file exists.

Return type bool

metadata_file_name = 'metadata.json'

mode
A read-only attribute representing the mode that the archive file was opened in.

color

This module provides functions for converting and using colors for arbitrary purposes including terminal output.

Functions

king_phisher.color.**convert_hex_to_tuple**(*hex_color*, *raw=False*)
Converts an RGB hex triplet such as #ff0000 into an RGB tuple. If *raw* is True then each value is on a scale from 0 to 255 instead of 0.0 to 1.0.

Parameters

- **hex_color** (*str*) – The hex code for the desired color.
- **raw** (*bool*) – Whether the values are raw or percentages.

Returns The color as a red, green, blue tuple.

Return type tuple

king_phisher.color.**convert_tuple_to_hex**(*rgb*, *raw=False*)
Converts an RGB color tuple info a hex string such as #ff0000. If *raw* is True then each value is treated as if it were on a scale from 0 to 255 instead of 0.0 to 1.0.

Parameters

- **rgb** (*tuple*) – The RGB tuple to convert into a string.
- **raw** (*bool*) – Whether the values are raw or percentages.

Returns The RGB color as a string.

Return type str

king_phisher.color.**get_scale**(*color_low*, *color_high*, *count*, *ascending=True*)
Create a scale of colors gradually moving from the low color to the high color.

Parameters

- **color_low** (*tuple*) – The darker color to start the scale with.
- **color_high** (*tuple*) – The lighter color to end the scale with.
- **count** – The total number of resulting colors.
- **ascending** (*bool*) – Whether the colors should be ascending from lighter to darker or the reverse.

Returns An array of colors starting with the low and gradually transitioning to the high.

Return type tuple

king_phisher.color.**print_error**(*message*)
Print an error message to the console.

Parameters **message** (*str*) – The message to print

king_phisher.color.**print_good**(*message*)
Print a good message to the console.

Parameters **message** (*str*) – The message to print

king_phisher.color.**print_status**(*message*)
Print a status message to the console.

Parameters **message** (*str*) – The message to print

Classes

class king_phisher.color.**ColoredLogFormatter**(*fmt=None*, *datefmt=None*)
A formatting class suitable for use with the logging module which colorizes the names of log levels.

format(*record*)

static formatException(*exc_info*)

constants

This module keeps collections of related constants organized for use in other modules.

Data

king_phisher.constants.**DEFAULT_LOG_LEVEL = 'WARNING'**
The default log level to use for filtering messages by importance.

king_phisher.constants.**DISABLED = Sentinel('DISABLED')**

Classes

class king_phisher.constants.**ConstantGroup**
A class for grouping related constants together.

classmethod items()
Iterate over the names and values in a group of constants.

classmethod names()
Iterate over the names in a group of constants.

classmethod values()
Iterate over the values in a group of constants.

class king_phisher.constants.**ConnectionErrorReason**
Constants which describe possible errors for an arbitrary connection process.

ConnectionErrorReason.**ERROR_AUTHENTICATION_FAILED = 'authentication failed'**

ConnectionErrorReason.**ERROR_CONNECTION = 'connection error'**

ConnectionErrorReason.**ERROR_INCOMPATIBLE_VERSIONS = 'incompatible versions'**

ConnectionErrorReason.**ERROR_INVALID_CREDENTIALS = 'invalid credentials'**

ConnectionErrorReason.**ERROR_INVALID_OTP = 'invalid otp'**

ConnectionErrorReason.**ERROR_PORT_FORWARD = 'port forward error'**

ConnectionErrorReason.**ERROR_UNKNOWN = 'unknown error'**

ConnectionErrorReason.**SUCCESS = 'success'**

class king_phisher.constants.**OSArch**
Constants for different operating system architectures.

OSArch.**PPC = 'PPC'**

OSArch.**X86 = 'x86'**

OSArch.**X86_64** = **'x86-64'**

class king_phisher.constants.**OSFamily**
Constants for families of different operating systems.

OSFamily.**ANDROID** = **'Android'**

OSFamily.**BLACKBERRY** = **'BlackBerry'**

OSFamily.**IOS** = **'iOS'**

OSFamily.**LINUX** = **'Linux'**

OSFamily.**OSX** = **'OS X'**

OSFamily.**WINDOWS** = **'Windows NT'**

errors

This module provides the custom exceptions that are used throughout the package.

Exceptions

exception king_phisher.errors.**KingPhisherError**(*message=''*)
Bases: exceptions.Exception

The base exception that is inherited by all custom King Phisher error classes.

exception king_phisher.errors.**KingPhisherAbortError**(*message=''*)
Bases: *king_phisher.errors.KingPhisherError*

An exception that can be raised to indicate that an arbitrary operation needs to be aborted when no better method can be used.

exception king_phisher.errors.**KingPhisherAbortRequestError**(*response_sent=False*)
Bases: *king_phisher.errors.KingPhisherAbortError*

An exception that can be raised which when caught will cause the handler to immediately stop processing the current request.

__init__(*response_sent=False*)

Parameters **response_sent** (*bool*) – Whether or not a response has already been sent to the client.

exception king_phisher.errors.**KingPhisherDatabaseError**(*message=''*)
Bases: *king_phisher.errors.KingPhisherError*

An exception that can be raised by King Phisher when there is any error relating to the database, it's configuration or any action involving it. The underlying database API will raise exceptions of it's own kind.

exception king_phisher.errors.**KingPhisherInputValidationError**(*message=''*)
Bases: *king_phisher.errors.KingPhisherError*

An exception that is raised when any kind of input into King Phisher fails to be properly validated.

exception king_phisher.errors.**KingPhisherPermissionError**(*message=''*)
Bases: *king_phisher.errors.KingPhisherError*

An exception that is raised by King Phisher when some form of a request can not be satisfied due to the configured level of access.

exception king_phisher.errors.**KingPhisherPluginError**(*plugin_name*, **args*, ***kwargs*)
Bases: *king_phisher.errors.KingPhisherError*

An exception that is raised by King Phisher to indicate an error regarding a particular plugin.

exception king_phisher.errors.**KingPhisherResourceError**(*message=''*)
Bases: *king_phisher.errors.KingPhisherError*

An exception that is raised by King Phisher when there is a problem relating to a resource such as it is missing, locked or otherwise invalid.

exception king_phisher.errors.**KingPhisherTimeoutError**(*message=''*)
Bases: *king_phisher.errors.KingPhisherError*

An exception that is raised by King Phisher when some form of a request fails to complete within a specified time period.

find

This module provides a means by which data files distributed with the application can be found at run time by searching a configurable set of directories.

Data

king_phisher.find.**DATA_DIRECTORY_NAME = 'king_phisher'**
The name of the directory containing the King Phisher data.

king_phisher.find.**ENV_VAR = 'KING_PHISHER_DATA_PATH'**
The name of the environment variable which contains the search path.

Functions

king_phisher.find.**data_path_append**(*path*)
Add a directory to the data search path. The directory will be used by the *data_file()* and *data_directory()* functions.

Parameters **path** (*str*) – The path to add for searching.

king_phisher.find.**data_directory**(*name*)
Locate a subdirectory in the data search path.

Parameters **name** (*str*) – The directory name to locate.

Returns The path to the located directory.

Return type str

king_phisher.find.**data_file**(*name*, *access_mode=4*)
Locate a data file by searching the directories specified in *ENV_VAR*. If *access_mode* is specified, it needs to be a value suitable for use with os.access().

Parameters

- **name** (*str*) – The name of the file to locate.
- **access_mode** (*int*) – The access that is required for the file.

Returns The path to the located file.

Return type str

king_phisher.find.**init_data_path**(*directory*)
Add a directory to the data search path for either client or server data files.

Parameters **directory** (*str*) – The directory to add, either 'client' or 'server'.

geoip

This module uses GeoLite2 data created by MaxMind, available from http://www.maxmind.com.

Data

king_phisher.geoip.**DB_DOWNLOAD_URL**
The URL from which the GeoLite2 City database can be downloaded from.

king_phisher.geoip.**DB_RESULT_FIELDS**
A tuple listing the fields that are required in database results.

Functions

king_phisher.geoip.**download_geolite2_city_db**(*dest*)
Download the GeoLite2 database, decompress it, and save it to disk.

Parameters **dest** (*str*) – The file path to save the database to.

king_phisher.geoip.**init_database**(*database_file*)
Create and initialize the GeoLite2 database engine. This must be done before classes and functions in this module attempt to look up results. If the specified database file does not exist, a new copy will be downloaded.

Parameters **database_file** (*str*) – The GeoLite2 database file to use.

Returns The initialized GeoLite2 database object.

Return type geoip2.database.Reader

king_phisher.geoip.**lookup**(*ip*, *lang='en'*)
Lookup the geo location information for the specified IP from the configured GeoLite2 City database.

Parameters

- **ip** (*str*) – The IP address to look up the information for.
- **lang** (*str*) – The language to prefer for regional names.

Returns The geo location information as a dict. The keys are the values of *DB_RESULT_FIELDS*.

Return type dict

Classes

class king_phisher.geoip.**Coordinates**(*latitude*, *longitude*)
A named tuple for representing GPS coordinates.

latitude
Alias for field number 0

longitude
Alias for field number 1

class king_phisher.geoip.**GeoLocation**(*ip*, *lang='en'*, *result=None*)
The geographic location information for a given IP address. If *result* is not specified, *lookup()* will be used to obtain the information.

__geo_interface__
A simple implementation of the Python __geo_interface__ specification. This allows this object to be used with modules which also support this interface such as geojson.

Returns A dictionary describing a this location as a GeoJSON Point.

Return type dict

__init__(*ip*, *lang='en'*, *result=None*)

Parameters

- **ip** (*str*) – The IP address to look up geographic location data for.
- **lang** (*str*) – The language to prefer for regional names.
- **result** (*dict*) – A raw query result from a previous call to *lookup()*.

city

continent

coordinates

country

ip_address
The IPv4Address which this geographic location data describes.

postal_code

time_zone

ics

This module provides functionality for creating **RFC 5545** compliant iCalendar invite files.

Data

king_phisher.ics.**DAY_ABBREVIATIONS**
The abbreviations of day names for use in icalendar.vRecur instances.

king_phisher.ics.**zoneinfo_path**
The path to the directory which holds the IANA timezone data files.

Functions

king_phisher.ics.**get_timedelta_for_offset**(*offset*)
Take a POSIX environment variable style offset from UTC and convert it into a timedelta instance suitable for use with the icalendar.

Parameters **offset** (*str*) – The offset from UTC such as "-5:00"

Returns The parsed offset.

Return type datetime.timedelta

king_phisher.ics.**get_tz_posix_env_var**(*tz_name*)

Get the timezone information in the POSIX TZ environment variable format from the IANA timezone data files included in the pytz package.

Parameters **tz_name** (*str*) – The name of the timezone to get the environment variable for such as "America/New_York".

Returns The TZ environment variable string, if it is specified in the timezone data file.

Return type str

king_phisher.ics.**parse_tz_posix_env_var**(*posix_env_var*)

Get the details regarding a timezone by parsing the POSIX style TZ environment variable.

Parameters **posix_env_var** (*str*) – The POSIX style TZ environment variable.

Returns The parsed TZ environment variable.

Return type *TimezoneOffsetDetails*

Classes

class king_phisher.ics.**Calendar**(*organizer_email*, *start*, *summary*, *organizer_cn=None*, *description=None*, *duration='1h'*, *location=None*)

Bases: icalendar.cal.Calendar

An icalendar formatted event for converting to an ICS file and then sending in an email.

__init__(*organizer_email*, *start*, *summary*, *organizer_cn=None*, *description=None*, *duration='1h'*, *location=None*)

Parameters

- **organizer_email** (*str*) – The email of the event organizer.
- **start** (datetime.datetime) – The start time for the event.
- **summary** (*str*) – A short summary of the event.
- **organizer_cn** (*str*) – The name of the event organizer.
- **description** (*str*) – A more complete description of the event than what is provided by the *summary* parameter.
- **duration** (int, str, timedelta, *DurationAllDay*) – The events scheduled duration.
- **location** (*str*) – The location for the event.

add_attendee(*email*, *cn=None*, *rsvp=True*)

Add an attendee to the event. If the event is being sent via an email, the recipient should be added as an attendee.

Parameters

- **email** (*str*) – The attendee's email address.
- **cn** (*str*) – The attendee's common name.
- **rsvp** (*bool*) – Whether or not to request an RSVP response from the attendee.

class king_phisher.ics.**DurationAllDay**(*days=1*)
Bases: object

A representation of a duration that can be used for an event to indicate that it takes place all day.

class king_phisher.ics.**Timezone**(*tz_name=None*)
Bases: icalendar.cal.Timezone

An icalendar formatted timezone with all properties populated for the specified zone.

__init__(*tz_name=None*)

Parameters **tz_name** (*str*) – The timezone to represent, if not specified it defaults to the local timezone.

class king_phisher.ics.**TimezoneOffsetDetails**(*offset*, *offset_dst*, *dst_start*, *dst_end*)
Bases: tuple

A named tuple describing the details of a timezone's UTC offset and DST occurrence.

dst_end
Alias for field number 3

dst_start
Alias for field number 2

offset
Alias for field number 0

offset_dst
Alias for field number 1

ipaddress

This module provides functionality for dealing with an external "ipaddress" module in a Python 2 backwards compatible way. In Python 2 all string address arguments are converted to unicode which removes the ability to specify addresses as packed binary strings.

Functions

king_phisher.ipaddress.**ip_address**(*address*, **args*, ***kwargs*)
Take an IP string/int and return an object of the correct type.

Args:

address: A string or integer, the IP address. Either IPv4 or IPv6 addresses may be supplied; integers less than 2**32 will be considered to be IPv4 by default.

Returns: An IPv4Address or IPv6Address object.

Raises:

ValueError: if the *address* passed isn't either a v4 or a v6 address

king_phisher.ipaddress.**ip_network**(*address*, **args*, ***kwargs*)
Take an IP string/int and return an object of the correct type.

Args:

address: A string or integer, the IP network. Either IPv4 or IPv6 networks may be supplied; integers less than 2**32 will be considered to be IPv4 by default.

Returns: An IPv4Network or IPv6Network object.

Raises:

ValueError: if the string passed isn't either a v4 or a v6 address. Or if the network has host bits set.

king_phisher.ipaddress.**ip_interface**(*address*, **args*, ***kwargs*)
Take an IP string/int and return an object of the correct type.

Args:

address: A string or integer, the IP address. Either IPv4 or IPv6 addresses may be supplied; integers less than 2**32 will be considered to be IPv4 by default.

Returns: An IPv4Interface or IPv6Interface object.

Raises:

ValueError: if the string passed isn't either a v4 or a v6 address.

Notes: The IPv?Interface classes describe an Address on a particular Network, so they're basically a combination of both the Address and Network classes.

king_phisher.ipaddress.**is_loopback**(*address*)
Check if an address is a loopback address or a common name for the loopback interface.

Parameters **address** (*str*) – The address to check.

Returns Whether or not the address is a loopback address.

Return type bool

king_phisher.ipaddress.**is_valid**(*address*, **args*, ***kwargs*)
Check that the string specified appears to be either a valid IPv4 or IPv6 address.

Parameters **address** (*str*) – The IP address to validate.

Returns Whether the IP address appears to be valid or not.

Return type bool

Classes

class king_phisher.ipaddress.**IPv4Address**(*address*)
Represent and manipulate single IPv4 Addresses.

is_link_local
Test if the address is reserved for link-local.

Returns: A boolean, True if the address is link-local per RFC 3927.

is_loopback
Test if the address is a loopback address.

Returns: A boolean, True if the address is a loopback per RFC 3330.

is_multicast
Test if the address is reserved for multicast use.

Returns: A boolean, True if the address is multicast. See RFC 3171 for details.

is_private
Test if this address is allocated for private networks.

Returns: A boolean, True if the address is reserved per iana-ipv4-special-registry.

is_reserved
Test if the address is otherwise IETF reserved.

Returns: A boolean, True if the address is within the reserved IPv4 Network range.

is_unspecified
Test if the address is unspecified.

Returns: A boolean, True if this is the unspecified address as defined in RFC 5735 3.

packed
The binary representation of this address.

class king_phisher.ipaddress.**IPv4Network**(*address*, *strict=True*)
This class represents and manipulates 32-bit IPv4 network + addresses..

Attributes: [examples for IPv4Network('192.0.2.0/27')] .network_address: IPv4Address('192.0.2.0') .hostmask: IPv4Address('0.0.0.31') .broadcast_address: IPv4Address('192.0.2.32') .netmask: IPv4Address('255.255.255.224') .prefixlen: 27

is_global
Test if this address is allocated for public networks.

Returns: A boolean, True if the address is not reserved per iana-ipv4-special-registry.

class king_phisher.ipaddress.**IPv6Address**(*address*)
Represent and manipulate single IPv6 Addresses.

ipv4_mapped
Return the IPv4 mapped address.

Returns: If the IPv6 address is a v4 mapped address, return the IPv4 mapped address. Return None otherwise.

is_global
Test if this address is allocated for public networks.

Returns: A boolean, true if the address is not reserved per iana-ipv6-special-registry.

is_link_local
Test if the address is reserved for link-local.

Returns: A boolean, True if the address is reserved per RFC 4291.

is_loopback
Test if the address is a loopback address.

Returns: A boolean, True if the address is a loopback address as defined in RFC 2373 2.5.3.

is_multicast
Test if the address is reserved for multicast use.

Returns: A boolean, True if the address is a multicast address. See RFC 2373 2.7 for details.

is_private
Test if this address is allocated for private networks.

Returns: A boolean, True if the address is reserved per iana-ipv6-special-registry.

is_reserved
Test if the address is otherwise IETF reserved.

Returns: A boolean, True if the address is within one of the reserved IPv6 Network ranges.

is_site_local
Test if the address is reserved for site-local.

Note that the site-local address space has been deprecated by RFC 3879. Use is_private to test if this address is in the space of unique local addresses as defined by RFC 4193.

Returns: A boolean, True if the address is reserved per RFC 3513 2.5.6.

is_unspecified
Test if the address is unspecified.

Returns: A boolean, True if this is the unspecified address as defined in RFC 2373 2.5.2.

packed
The binary representation of this address.

sixtofour
Return the IPv4 6to4 embedded address.

Returns: The IPv4 6to4-embedded address if present or None if the address doesn't appear to contain a 6to4 embedded address.

teredo
Tuple of embedded teredo IPs.

Returns: Tuple of the (server, client) IPs or None if the address doesn't appear to be a teredo address (doesn't start with 2001::/32)

class king_phisher.ipaddress.**IPv6Network**(*address*, *strict=True*)
This class represents and manipulates 128-bit IPv6 networks.

Attributes: [examples for IPv6('2001:db8::1000/124')] .network_address: IPv6Address('2001:db8::1000') .hostmask: IPv6Address('::f') .broadcast_address: IPv6Address('2001:db8::100f') .netmask: IPv6Address('ffff:ffff:ffff:ffff:ffff:ffff:ffff:fff0') .prefixlen: 124

hosts()
Generate Iterator over usable hosts in a network.

This is like __iter__ except it doesn't return the Subnet-Router anycast address.

is_site_local
Test if the address is reserved for site-local.

Note that the site-local address space has been deprecated by RFC 3879. Use is_private to test if this address is in the space of unique local addresses as defined by RFC 4193.

Returns: A boolean, True if the address is reserved per RFC 3513 2.5.6.

plugins

This module provides the core functionality necessary to support user provided plugins.

Classes

class king_phisher.plugins.**OptionBase**(*name*, *description*, *default=None*)
Bases: object

A base class for options which can be configured for plugins.

__init__(*name*, *description*, *default=None*)

Parameters

- **name** (*str*) – The name of this option.
- **description** (*str*) – The description of this option.
- **default** – The default value of this option.

class king_phisher.plugins.**OptionBoolean**(*name*, *description*, *default=None*)
Bases: *king_phisher.plugins.OptionBase*

A plugin option which is represented with a boolean value.

__init__(*name*, *description*, *default=None*)

Parameters

- **name** (*str*) – The name of this option.
- **description** (*str*) – The description of this option.
- **default** – The default value of this option.

class king_phisher.plugins.**OptionEnum**(*name*, *description*, *choices*, *default=None*)
Bases: *king_phisher.plugins.OptionBase*

A plugin option which is represented with an enumerable value.

__init__(*name*, *description*, *choices*, *default=None*)

Parameters

- **name** (*str*) – The name of this option.
- **description** (*str*) – The description of this option.
- **choices** (*tuple*) – The supported values for this option.
- **default** – The default value of this option.

class king_phisher.plugins.**OptionInteger**(*name*, *description*, *default=None*)
Bases: *king_phisher.plugins.OptionBase*

A plugin option which is represented with an integer value.

__init__(*name*, *description*, *default=None*)

Parameters

- **name** (*str*) – The name of this option.
- **description** (*str*) – The description of this option.
- **default** – The default value of this option.

class king_phisher.plugins.**OptionString**(*name*, *description*, *default=None*)
Bases: *king_phisher.plugins.OptionBase*

A plugin option which is represented with a string value.

__init__(*name*, *description*, *default=None*)

Parameters

- **name** (*str*) – The name of this option.
- **description** (*str*) – The description of this option.
- **default** – The default value of this option.

class king_phisher.plugins.**PluginBase**
Bases: *king_phisher.plugins.PluginBaseMeta*

A base class to be inherited by all plugins. Overriding or extending the standard __init__ method should be avoided to be compatible with future API changes. Instead the *initialize()* and *finalize()* methods should be overridden to provide plugin functionality.

authors = []
The list of authors who have provided this plugin.

config = None
The plugins configuration dictionary for storing the values of it's options.

description = None
A description of the plugin and what it does.

finalize()
This method can be overridden to perform any clean up action that the plugin needs such as closing files. It is called automatically by the manager when the plugin is disabled.

homepage = None
An optional homepage for the plugin.

initialize()
This method should be overridden to provide the primary functionality of the plugin. It is called automatically by the manager when the plugin is enabled.

Returns Whether or not the plugin successfully initialized itself.

Return type bool

options = []
A list of configurable option definitions for the plugin.

req_min_version = '1.3.0b0'
The required minimum version for compatibility.

req_packages = {}
A dictionary of required packages, keyed by the package name and a boolean value of it's availability.

title = None
The title of the plugin.

version = '1.0'
The version identifier of this plugin.

class king_phisher.plugins.**PluginBaseMeta**
Bases: type

The meta class for *PluginBase* which provides additional class properties based on defined attributes.

compatibility
A generator which yields tuples of compatibility information based on the classes defined attributes. Each tuple contains three elements, a string describing the requirement, the requirements value, and a boolean indicating whether or not the requirement is met.

Returns Tuples of compatibility information.

formatted_description
Format the text description by removing leading whitespace caused by the definition within the class.

is_compatible
Whether or not this plugin is compatible with this version of King Phisher. This can only be checked after

the module is imported, so any references to non-existent classes in older versions outside of the class methods will still cause a load error.

Returns Whether or not this plugin class is compatible.

Return type bool

class king_phisher.plugins.**PluginManagerBase**(*path*, *args=None*)
Bases: object

A managing object to control loading and enabling individual plugin objects.

__init__(*path*, *args=None*)

Parameters

- **path** (*tuple*) – A tuple of directories from which to load plugins.
- **args** (*tuple*) – Arguments which should be passed to plugins when their class is initialized.

available
Return a tuple of all available plugins that can be loaded.

disable(*name*)
Disable a plugin by it's name. This call the plugins *PluginBase.finalize()* method to allow it to perform any clean up operations.

Parameters **name** (*str*) – The name of the plugin to disable.

enable(*name*)
Enable a plugin by it's name. This will create a new instance of the plugin modules "Plugin" class, passing it the arguments defined in plugin_init_args. A reference to the plugin instance is kept in *enabled_plugins*. After the instance is created, the plugins *initialize()* method is called.

Parameters **name** (*str*) – The name of the plugin to enable.

Returns The newly created instance.

Return type *PluginBase*

enabled_plugins = None
A dictionary of the enabled plugins and their respective instances.

load(*name*, *reload_module=False*)
Load a plugin into memory, this is effectively the Python equivalent of importing it. A reference to the plugin class is kept in *loaded_plugins*. If the plugin is already loaded, no changes are made.

Parameters

- **name** (*str*) – The name of the plugin to load.
- **reload_module** (*bool*) – Reload the module to allow changes to take affect.

Returns

load_all(*on_error=None*)
Load all available plugins. Exceptions while loading specific plugins are ignored. If *on_error* is specified, it will be called from within the exception handler when a plugin fails to load correctly. It will be called with two parameters, the name of the plugin and the exception instance.

Parameters **on_error** (*function*) – A call back function to call when an error occurs while loading a plugin.

loaded_plugins = None
A dictionary of the loaded plugins and their respective modules.

shutdown()
Unload all plugins and perform additional clean up operations.

unload(*name*)
Unload a plugin from memory. If the specified plugin is currently enabled, it will first be disabled before being unloaded. If the plugin is not already loaded, no changes are made.

Parameters **name** (*str*) – The name of the plugin to unload.

unload_all()
Unload all available plugins. Exceptions while unloading specific plugins are ignored.

scrubber

This module provides functionality for removing metadata from certain types of files.

Functions

king_phisher.scrubber.**remove_office_metadata**(*file_name*)
Remove all metadata from Microsoft Office 2007+ file types such as docx, pptx, and xlsx.

Parameters **file_name** (*str*) – The path to the file whose metadata is to be removed.

serializers

This module provides a standardized interface for serializing objects using different formats. The Serializers provided by this module are organized by their format into different classes. The necessary methods for utilizing them are all `classmethod`'s making it unnecessary to create an instance of any of them.

Functions

king_phisher.serializers.**from_elementtree_element**(*element*, *require_type=True*)
Load a value from an `xml.etree.ElementTree.SubElement` instance. If *require_type* is True, then the element must specify an acceptable value via the "type" attribute. If *require_type* is False and no type attribute is specified, the value is returned as a string.

Parameters

- **element** (`xml.etree.ElementTree.Element`) – The element to load a value from.
- **require_type** (*bool*) – Whether or not to require type information.

Returns The deserialized value from the element.

king_phisher.serializers.**to_elementtree_subelement**(*parent*, *tag*, *value*, *attrib=None*)
Serialize *value* to an `xml.etree.ElementTree.SubElement` with appropriate information describing it's type. If *value* is not of a supported type, a `TypeError` will be raised.

Parameters

- **parent** (`xml.etree.ElementTree.Element`) – The parent element to associate this subelement with.
- **tag** (*str*) – The name of the XML tag.

- **value** – The value to serialize to an XML element.
- **attrib** (*dict*) – Optional attributes to include in the element.

Returns The newly created XML element, representing *value*.

Return type xml.etree.ElementTree.Element

Classes

class king_phisher.serializers.**JSON**
Bases: *king_phisher.serializers.Serializer*

classmethod dumps(*data*, *pretty=True*)
Convert a Python object to a JSON encoded string.

Parameters

- **data** – The object to encode.
- **pretty** (*bool*) – Set options to make the resulting JSON data more readable.

Returns The encoded data.

Return type str

classmethod loads(*data*, *strict=True*)
Load JSON encoded data.

Parameters

- **data** (*str*) – The encoded data to load.
- **strict** (*bool*) – Do not try remove trailing commas from the JSON data.

Returns The Python object represented by the encoded data.

class king_phisher.serializers.**MsgPack**
Bases: *king_phisher.serializers.Serializer*

classmethod dumps(*data*)
Convert a Python object to a MsgPack encoded bytes instance.

Parameters

- **data** – The object to encode.
- **pretty** (*bool*) – Set options to make the resulting JSON data more readable.

Returns The encoded data.

Return type str

classmethod loads(*data*)
Load MsgPack encoded data.

Parameters **data** (*bytes*) – The encoded data to load.

Returns The Python object represented by the encoded data.

class king_phisher.serializers.**Serializer**
Bases: king_phisher.serializers._SerializerMeta

The base class for serializer objects of different formats and protocols. These serializers are extended using a King Phisher-specific protocol for serializing additional types, most notably Python's datetime.datetime type.

Note: None of the serializers handle Python 3's `bytes` type. These objects will be treated as strings and silently converted.

classmethod dump(*data*, *file_h*, **args*, ***kwargs*)
Write a Python object to a file by encoding it with this serializer.

Parameters

- **data** – The object to encode.
- **file_h** (*file*) – The file to write the encoded string to.

encoding = 'utf-8'
The encoding which this serializer uses for handling strings.

classmethod load(*file_h*, **args*, ***kwargs*)
Load encoded data from the specified file.

Parameters

- **file_h** (*file*) – The file to read and load encoded data from.
- **strict** (*bool*) – Do not try remove trailing commas from the JSON data.

Returns The Python object represented by the encoded data.

sms

This module provides functionality for sending free SMS messages by emailing a carriers SMS gateway.

Data

king_phisher.sms.**CARRIERS**
A dictionary for mapping carrier names to SMS via email gateways.

king_phisher.sms.**DEFAULT_FROM_ADDRESS**
The default email address to use in the from field.

Functions

king_phisher.sms.**get_smtp_servers**(*domain*)
Get the SMTP servers for the specified domain by querying their MX records.

Parameters **domain** (*str*) – The domain to look up the MX records for.

Returns The smtp servers for the specified domain.

Return type list

king_phisher.sms.**lookup_carrier_gateway**(*carrier*)
Lookup the SMS gateway for the specified carrier. Normalization on the carrier name does take place and if an invalid or unknown value is specified, None will be returned.

Parameters **carrier** (*str*) – The name of the carrier to lookup.

Returns The SMS gateway for the specified carrier.

Return type str

king_phisher.sms.**send_sms**(*message_text*, *phone_number*, *carrier*, *from_address=None*)
Send an SMS message by emailing the carriers SMS gateway. This method requires no money however some networks are blocked by the carriers due to being flagged for spam which can cause issues.

Parameters

- **message_text** (*str*) – The message to send.
- **phone_number** (*str*) – The phone number to send the SMS to.
- **carrier** (*str*) – The cellular carrier that the phone number belongs to.
- **from_address** (*str*) – The optional address to display in the 'from' field of the SMS.

Returns This returns the status of the sent messsage.

Return type bool

smtp_server

This module provides a SMTP server that can be used for debugging purposes.

Classes

class king_phisher.smtp_server.**BaseSMTPServer**(*localaddr*, *remoteaddr=None*)
Bases: smtpd.SMTPServer, object

An SMTP server useful for debugging. Messages handled by this server are not forwarded anywhere.

__init__(*localaddr*, *remoteaddr=None*)

Parameters

- **localaddr** (*tuple*) – The local address to bind to.
- **remoteaddr** (*tuple*) – The remote address to use as an upstream SMTP relayer.

serve_forever()
Process requests until a BaseSMTPServer.shutdown() is called.

spf

This module provides functionality for checking published Sender Policy Framework (SPF) records. SPF is defined in **RFC 7208**.

Data

king_phisher.spf.**MACRO_REGEX**
A regular expression which matches SPF record macros.

king_phisher.spf.**MAX_QUERIES = 10**
The maximum number of DNS queries allowed to take place during evaluation as defined within section 4.6.4 of **RFC 7208**.

king_phisher.spf.**MAX_QUERIES_VOID = inf**
The maximum number of DNS queries allowed to either return with rcode 0 and no answers or rcode 3 (Name Error) as defined within section 4.6.4 of **RFC 7208**.

king_phisher.spf.**QUALIFIERS**
A dict object keyed with the qualifier symbols to their readable values.

Functions

king_phisher.spf.**check_host**()
Analyze the Sender Policy Framework of a domain by creating a *SenderPolicyFramework* instance and returning the result of *SenderPolicyFramework.check_host()*.

Parameters

- **ip** (str, ipaddress.IPv4Address, ipaddress.IPv6Address) – The IP address of the host sending the message.
- **domain** (*str*) – The domain to check the SPF policy of.
- **sender** (*str*) – The "MAIL FROM" identity of the message being sent.

Returns The result of the SPF policy if one can be found or None.

Return type None, str

king_phisher.spf.**validate_record**(*ip*, *domain*, *sender=None*)
Check if an SPF record exists for the domain and can be parsed by this module.

Returns Whether the record exists and is parsable or not.

Return type bool

Classes

class king_phisher.spf.**SenderPolicyFramework**(*ip*, *domain*, *sender=None*)
Analyze the Sender Policy Framework configuration for a domain to determine if an IP address is authorized to send messages on it's behalf. The exp modifier defined in section 6.2 of the RFC is not supported.

__init__(*ip*, *domain*, *sender=None*)

Parameters

- **ip** (str, ipaddress.IPv4Address, ipaddress.IPv6Address) – The IP address of the host sending the message.
- **domain** (*str*) – The domain to check the SPF policy of.
- **sender** (*str*) – The "MAIL FROM" identity of the message being sent.

check_host()
Check the SPF policy described by the object. The string representing the matched policy is returned if an SPF policy exists, otherwise None will be returned if no policy is defined.

Returns The result of the SPF policy described by the object.

Return type None, str

expand_macros(*value*, *ip*, *domain*, *sender*)
Expand a string based on the macros it contains as specified by section 7 of **RFC 7208**.

Parameters

- **value** (*str*) – The string containing macros to expand.
- **ip** (str, ipaddress.IPv4Address, ipaddress.IPv6Address) – The IP address to use when expanding macros.
- **domain** (*str*) – The domain name to use when expanding macros.
- **sender** (*str*) – The email address of the sender to use when expanding macros.

Returns The string with the interpreted macros replaced within it.

Return type str

match

matches = None
A list of *SPFMatch* instances showing the path traversed to identify a matching directive. Multiple entries in this list are present when include directives are used and a match is found within the body of one. The list is ordered from the top level domain to the matching record.

policy = None
The human readable policy result, one of the SPFResult constants‘.

records = None
A collections.OrderedDict of all the SPF records that were resolved. This would be any records resolved due to an “include” directive in addition to the top level domain.

class king_phisher.spf.**SPFDirective**(*mechanism*, *qualifier*, *rvalue=None*)
A class representing a single directive within a sender policy framework record.

class king_phisher.spf.**SPFMatch**(*record*, *directive*)
A simple container to associate a matched directive with it’s record.

class king_phisher.spf.**SPFRecord**(*directives*, *domain=None*)
A class representing a parsed Sender Policy Framework record with all of its directives.

Exceptions

exception king_phisher.spf.**SPFError**(*message*)
Bases: exceptions.Exception

Base exception for errors raised by this module.

exception king_phisher.spf.**SPFTempError**(*message*)
Bases: *king_phisher.spf.SPFError*

Exception indicating that the verification process encountered a transient (generally DNS) error while performing the check. Described in section 2.6.6 of **RFC 7208**.

exception king_phisher.spf.**SPFParseError**(*message*)
Bases: *king_phisher.spf.SPFPermError*

Exception indicating that the domains published records could not be correctly parsed.

exception king_phisher.spf.**SPFPermError**(*message*)
Bases: *king_phisher.spf.SPFError*

Exception indicating that the domains published records could not be correctly interpreted. Described in section 2.6.7 of **RFC 7208**.

ssh_forward

This module provides functionality for forwarding network services over SSH.

Classes

class king_phisher.ssh_forward.**SSHTCPForwarder**(*server*, *username*, *password*, *remote_server*, *local_port=0*, *private_key=None*, *missing_host_key_policy=None*)

Bases: threading.Thread

Open an SSH connection and forward TCP traffic through it to a remote host. A private key for authentication can be specified as a string either by it's OpenSSH fingerprint, as a file (prefixed with "file:"), or a raw key string (prefixed with "key:"). If no *missing_host_key_policy* is specified, paramiko.client.AutoAddPolicy will be used to accept all host keys.

Note: This is a threading.Thread object and needs to be started with a call to start() after it is initialized.

__init__(*server*, *username*, *password*, *remote_server*, *local_port=0*, *private_key=None*, *missing_host_key_policy=None*)

Parameters

- **server** (*tuple*) – The SSH server to connect to.
- **username** (*str*) – The username to authenticate with.
- **password** (*str*) – The password to authenticate with.
- **remote_server** (*tuple*) – The remote server to connect to through the specified SSH server.
- **local_port** (*int*) – The local port to forward, if not set a random one will be used.
- **private_key** (*str*) – An RSA key to prefer for authentication.
- **missing_host_key_policy** – The policy to use for missing host keys.

local_server

A tuple representing the local address of the listening service which is forwarding traffic to the specified remote host.

Exceptions

class king_phisher.ssh_forward.**KingPhisherSSHKeyError**(*message=''*)

Bases: *king_phisher.errors.KingPhisherError*

An exception that is thrown when there is a problem resolving a users SSH key file. The *message* attribute is formatted to be displayed to the user via a dialog.

templates

This module provides base classes for the Jinja environments used throughout the application.

Classes

class king_phisher.templates.**TemplateEnvironmentBase**(*loader=None*, *global_vars=None*)
Bases: jinja2.environment.Environment

A configured Jinja2 environment with additional filters.

__init__(*loader=None*, *global_vars=None*)

Parameters

- **loader** (jinja2.BaseLoader) – The loader to supply to the environment.
- **global_vars** (*dict*) – Additional global variables for the environment.

join_path(*template*, *parent*)
Over ride the default jinja2.Environment.join_path() method to explicitly specifying relative paths by prefixing the path with either "./" or "../".

Parameters

- **template** (*str*) – The path of the requested template file.
- **parent** (*str*) – The path of the template file which requested the load.

Returns The new path to the template.

Return type str

standard_variables
Additional standard variables that can optionally be used for templates.

class king_phisher.templates.**MessageTemplateEnvironment**(**args*, ***kwargs*)
Bases: *king_phisher.templates.TemplateEnvironmentBase*

A configured Jinja2 environment for formatting messages.

MODE_ANALYZE = 1

MODE_PREVIEW = 0

MODE_SEND = 2

attachment_images = None
A dictionary collecting the images that are going to be embedded and sent inline in the message.

set_mode(*mode*)
Set the operation mode for the environment. Valid values are the MODE_* constants.

Parameters **mode** (*int*) – The operation mode.

testing

This module provides supporting functionality for the included application unit tests.

Data

king_phisher.testing.**TEST_MESSAGE_TEMPLATE**
A string representing a message template that can be used for testing.

king_phisher.testing.**TEST_MESSAGE_TEMPLATE_INLINE_IMAGE**
A string with the path to a file used as an inline image in the *TEST_MESSAGE_TEMPLATE*.

Classes

class king_phisher.testing.**KingPhisherTestCase**(**args*, ***kwargs*)
Bases: smoke_zephyr.utilities.TestCase

This class provides additional functionality over the built in unittest.TestCase object, including better compatibility for methods across Python 2.x and Python 3.x.

class king_phisher.testing.**KingPhisherServerTestCase**(**args*, ***kwargs*)
Bases: *king_phisher.testing.KingPhisherTestCase*

This class can be inherited to automatically set up a King Phisher server instance configured in a way to be suitable for testing purposes.

assertHTTPStatus(*http_response*, *status*)
Check an HTTP response to ensure that the correct HTTP status code is specified.

Parameters

- **http_response** (httplib.HTTPResponse) – The response object to check.
- **status** (*int*) – The status to check for.

assertRPCPermissionDenied(*method*, **args*, ***kwargs*)
Assert that the specified RPC method fails with a *KingPhisherPermissionError* exception.

Parameters **method** – The RPC method that is to be tested

http_request(*resource*, *method='GET'*, *include_id=True*, *body=None*, *headers=None*)
Make an HTTP request to the specified resource on the test server.

Parameters

- **resource** (*str*) – The resource to send the request to.
- **method** (*str*) – The HTTP method to use for the request.
- **include_id** (*bool*) – Whether to include the the id parameter.
- **body** (*dict, str*) – The data to include in the body of the request.
- **headers** (*dict*) – The headers to include in the request.

Returns The servers HTTP response.

Return type httplib.HTTPResponse

web_root_files(*limit=None*, *include_templates=True*)
A generator object that yields valid files which are contained in the web root of the test server instance. This can be used to find resources which the server should process as files. The function will fail if no files can be found in the web root.

Parameters

- **limit** (*int*) – A limit to the number of files to return.
- **include_templates** (*bool*) – Whether or not to include files that might be templates.

ua_parser

This module provides functionality for parsing browser user agents to extract information from them.

Functions

king_phisher.ua_parser.**parse_user_agent**(*user_agent*)
Parse a user agent string and return normalized information regarding the operating system.

Parameters **user_agent** (*str*) – The user agent to parse.

Returns A parsed user agent, None is returned if the data can not be processed.

Return type *UserAgent*

Classes

class king_phisher.ua_parser.**UserAgent**
A parsed representation of the information available from a browsers user agent string. Only the *os_name* attribute is guaranteed to not be None.

os_name
The *OSFamily* constant of the name of the operating system.

os_version
The version of the operating system.

os_arch
The *OSArch* constant of the architecture of the operating system.

utilities

This module collects various useful utility functions that are used throughout the application.

Functions

king_phisher.utilities.**argp_add_args**(*parser*, *default_root=''*)
Add standard arguments to a new argparse.ArgumentParser instance for configuring logging options from the command line and displaying the version information.

Note: This function installs a hook to *parser.parse_args* to automatically handle options which it adds. This includes setting up a stream logger based on the added options.

Parameters

- **parser** (argparse.ArgumentParser) – The parser to add arguments to.
- **default_root** (*str*) – The default root logger to specify.

king_phisher.utilities.**assert_arg_type**(*arg*, *arg_type*, *arg_pos=1*, *func_name=None*)
Check that an argument is an instance of the specified type and if not raise a TypeError exception with a meaningful message. If *func_name* is not specified, it will be determined by examining the stack.

Parameters

- **arg** – The argument to check.
- **arg_type** (*list, tuple, type*) – The type or sequence of types that *arg* can be.

- **arg_pos** (*int*) – The position of the argument in the function.
- **func_name** (*str*) – The name of the function the argument is for.

king_phisher.utilities.**configure_stream_logger**(*logger*, *level=None*)
Configure the default stream handler for logging messages to the console. This also configures the basic logging environment for the application.

Parameters

- **logger** (*str*) – The logger to add the stream handler for.
- **level** (*None, int, str*) – The level to set the logger to, will default to WARNING if no level is specified.

Returns The new configured stream handler.

Return type logging.StreamHandler

king_phisher.utilities.**datetime_local_to_utc**(*dt*)
Convert a datetime.datetime instance from the local time to UTC time.

Parameters **dt** (datetime.datetime) – The time to convert from local to UTC.

Returns The time converted to the UTC timezone.

Return type datetime.datetime

king_phisher.utilities.**datetime_utc_to_local**(*dt*)
Convert a datetime.datetime instance from UTC time to the local time.

Parameters **dt** (datetime.datetime) – The time to convert from UTC to local.

Returns The time converted to the local timezone.

Return type datetime.datetime

king_phisher.utilities.**format_datetime**(*dt*, *encoding='utf-8'*)
Format a date time object into a string. If the object *dt* is not an instance of datetime.datetime then an empty string will be returned.

Parameters

- **dt** (datetime.datetime) – The object to format.
- **encoding** (*str*) – The encoding to use to coerce the return value into a unicode string.

Returns The string representing the formatted time.

Return type str

king_phisher.utilities.**is_valid_email_address**(*email_address*)
Check that the string specified appears to be a valid email address.

Parameters **email_address** (*str*) – The email address to validate.

Returns Whether the email address appears to be valid or not.

Return type bool

king_phisher.utilities.**make_message_uid**()
Creates a random string of characters and numbers to be used as a message id.

Returns String of characters from the random_string function.

Return type str

king_phisher.utilities.**make_visit_uid**()
Creates a random string of characters and numbers to be used as a visit id.

Returns String of characters from the random_string function.

Return type str

king_phisher.utilities.**open_uri**(*uri*)
Open a URI in a platform intelligent way. On Windows this will use 'cmd.exe /c start' and on Linux this will use gvfs-open or xdg-open depending on which is available. If no suitable application can be found to open the URI, a RuntimeError will be raised.

Parameters **uri** (*str*) – The URI to open.

king_phisher.utilities.**parse_datetime**(*ts*)
Parse a time stamp into a datetime.datetime instance. The time stamp must be in a compatible format, as would have been returned from the *format_datetime()* function.

Parameters **ts** (*str*) – The timestamp to parse.

Returns The parsed timestamp.

Return type datetime.datetime

king_phisher.utilities.**password_is_complex**(*password*, *min_len=12*)
Check that the specified string meets standard password complexity requirements. :param str password: The password to validate. :param int min_len: The mininum length the password should be. :return: Whether the strings appears to be complex or not. :rtype: bool

king_phisher.utilities.**random_string**(*size*)
Generate a random string consisting of uppercase letters, lowercase letters and numbers of the specified size.

Parameters **size** (*int*) – The size of the string to make.

Returns The string containing the random characters.

Return type str

king_phisher.utilities.**random_string_lower_numeric**(*size*)
Generate a random string consisting of lowercase letters and numbers of the specified size.

Parameters **size** (*int*) – The size of the string to make.

Returns The string containing the random characters.

Return type str

king_phisher.utilities.**start_process**(*proc_args*, *wait=True*)
Start an external process.

Parameters

- **proc_args** (*list, str*) – The arguments of the process to start.
- **wait** (*bool*) – Wait for the process to exit before returning.

king_phisher.utilities.**switch**(*value*, *comp=<built-in function eq>*, *swapped=False*)
A pure Python implementation of a switch case statement. *comp* will be used as a comparison function and passed two arguments of *value* and the provided case respectively.

Switch case example usage:

```
for case in switch(2):
    if case(1):
        print('case 1 matched!')
        break
```

```
    if case(2):
        print('case 2 matched!')
        break
else:
    print('no cases were matched')
```

Parameters

- **value** – The value to compare in each of the case statements.
- **comp** – The function to use for comparison in the case statements.
- **swapped** – Whether or not to swap the arguments to the *comp* function.

Returns A function to be called for each case statement.

Classes

class king_phisher.utilities.**FreezableDict**(**args*, ***kwargs*)

Bases: collections.OrderedDict

A dictionary that can be frozen to prevent further editing. Useful for debugging. If any function tries to edit a frozen dictionary, a RuntimeError will be raised and a traceback will occur.

freeze()

Freeze the dictionary to prevent further editing.

frozen

Whether or not the dictionary is frozen and can not be modified.

Return type bool

thaw()

Thaw the dictionary to once again enable editing.

class king_phisher.utilities.**Mock**(**args*, ***kwargs*)

Bases: object

A fake object used to replace missing imports when generating documentation.

version

This module collects all import version information for the application. This is the authoritative source for the applications version information and should be used anywhere the version is required.

Data

king_phisher.version.**distutils_version = '1.8.0b0'**

A string suitable for being parsed by distutils.version classes.

king_phisher.version.**revision = u'9a1e13b0e0029f841ea847d9848c3081182c3719'**

The git revision identifying the latest commit if available.

king_phisher.version.**rpc_api_version = rpc_api_version(major=5, minor=5)**

A tuple representing the local version of the RPC API for use with compatibility checks. The major version

is incremented when backwards incompatible changes are made and the minor version is incremented when backwards compatible changes are made.

king_phisher.version.**version = '1.8.0-beta (rev: 9a1e13b0e002)'**
A string representing the full version information.

king_phisher.version.**version_info = version_info(major=1, minor=8, micro=0)**
A tuple representing the version information in the format ('major', 'minor', 'micro')

king_phisher.version.**version_label = 'beta'**
A version label such as alpha or beta.

Functions

king_phisher.version.**get_revision**()
Retrieve the current git revision identifier. If the git binary can not be found or the repository information is unavailable, None will be returned.

Returns The git revision tag if it's available.

Return type str

xor

This module provides basic support for XOR encoding and decoding operations.

Functions

king_phisher.xor.**xor_decode**(*data*)
Decode data using the XOR algorithm. This is not suitable for encryption purposes and should only be used for light obfuscation. This function requires the key to be set as the first byte of *data* as done in the *xor_encode()* function.

Parameters **data** (*str*) – The data to decode.

Returns The decoded data.

Return type str

king_phisher.xor.**xor_encode**(*data*, *seed_key=None*)
Encode data using the XOR algorithm. This is not suitable for encryption purposes and should only be used for light obfuscation. The key is prepended to the data as the first byte which is required to be decoded py the *xor_decode()* function.

Parameters

- **data** (*str*) – The data to encode.
- **seed_key** (*int*) – The optional value to use as the for XOR key.

Returns The encoded data.

Return type str

CHAPTER 2

The King Phisher Client

Additional Configuration

The following configuration settings will be honored but can not be set from within the client's user interface.

Setting Name	Default Value
gui.refresh_frequency	5 minutes
mailer.max_messages_per_connection	5 messages
rpc.serializer	Automatically determined
ssh_preferred_key	Automatically determined

Completion Data

Some classes provided by the *widget.completion_providers* module require large amounts of data to function. This data is stored encoded in JSON to be loaded when these classes are initialized. The formats of the data are specific to each completion provider depending on the needs of their target syntax.

HTML

The HTML data file is a dictionary whose keys are HTML 5 tags such as body, input and script. Each of these keys values is either None if the tag does not have any attributes or a list of the valid attribute names. Each of the defined attributes are assumed to require a value, however ones which do not are suffixed with `!`. This suffix is used by the completion provider to determine if the opening definition for an attribute (`="`) should be appended to the token or not.

Example data containing completion information for the html and input tags:

```
{
  "html": null,
  "input": [
    "disabled!",
```

```
    "type"
   ]
}
```

Jinja

The Jinja data file is a dictionary containing two sub keys of `global` and `context` for global, and context specific data respectively. The global key's value is a dictionary containing three subkeys of `filters`, `tests` and `tokens` for the different kinds of Jinja terms which should be auto completed. The filters and tests keys have values of lists including all of the defined Jinja filters and tests respectively.

The tokens key has a value of a dictionary which contains the tokens broken out into a hierarchy of objects and attributes. Attributes which have sub-attributes are represented as dictionaries while attributes which have no attributes and are thus leaves have values of None. In the context of completion, variables and functions are treated as tokens because neither one are dependant on presence of a setup statement which is the case with filters and tests.

Tokens, filters and tests which are callable and require at least one argument to be specified are all suffixed with `(`. This suffix is used by the completion provider to signify that arguments are expected.

The top-level context key contains subkeys that define additional data to be merged with the global filters, tests and tokens based on a defined context. This allows the global Jinja environment data to be added to context specific providers.

Example data containing global filters, tests and tokens along with a truncated "email" context.

```
{
  "context": {
    "email": {
      "tokens": {
        ...
      }
    }
  },
  "global": {
    "filters": [
      "replace(",
      "title"
    ],
    "tests": [
      "defined",
      "equalto("
    ],
    "tokens": {
      "range(": null,
      "time": {
        "local": null,
        "utc": null
      },
      "version": null
    }
  }
}
```

GObject Signals

These signals can be used by the client API and plugins to subscribe to specific events. To explicitly connect after the default handler for a signal, use the *connect_after* method instead of *connect*. Some signals require a value to be returned by their handlers as noted.

Application Signals

The following are the signals for the *KingPhisherClientApplication* object.

campaign-changed(campaign_id)
This signal is emitted when campaign attributes are changed. Subscribers to this signal can use it to update and refresh information for the modified campaign.

Signal flags SIGNAL_RUN_FIRST

Parameters **campaign_id** (*str*) – The ID of the campaign whose information was changed.

campaign-created(campaign_id)
This signal is emitted after the user creates a new campaign id. Subscribers to this signal can use it to conduct an action after a new campaign id is created.

Signal flags SIGNAL_RUN_FIRST

Parameters **campaign_id** (*str*) – The ID of the new campaign.

campaign-delete(campaign_id)
This signal is emitted when the user deletes a campaign. Subscribers to this signal can use it to conduct an action after the campaign is deleted.

Signal flags SIGNAL_ACTION | SIGNAL_RUN_LAST

Parameters **campaign_id** (*str*) – The ID of the campaign.

campaign-set(old_campaign_id, new_campaign_id)
This signal is emitted when the user sets the current campaign. Subscribers to this signal can use it to update and refresh information for the current campaign. The `config` "campaign_id" and "campaign_name" keys have already been updated with the new values when this signal is emitted.

Signal flags SIGNAL_RUN_FIRST

Parameters

- **old_campaign_id** (*str*) – The ID of the old campaign or None if the client is selecting one for the first time.
- **new_campaign_id** (*str*) – The ID of the new campaign.

config-load(load_defaults)
This signal is emitted when the client configuration is loaded from disk. This loads all of the clients settings used within the GUI.

Signal flags SIGNAL_ACTION | SIGNAL_RUN_LAST

Parameters **load_defaults** (*bool*) – Load missing options from the template configuration file.

config-save()
This signal is emitted when the client configuration is written to disk. This saves all of the settings used within the GUI so they can be restored at a later point in time.

Signal flags SIGNAL_ACTION | SIGNAL_RUN_LAST

credential-delete(row_ids)
This signal is emitted when the user deletes a credential entry. Subscribers to this signal can use it to conduct an action an entry is deleted.

Signal flags SIGNAL_ACTION | SIGNAL_RUN_LAST

Parameters **row_ids** (*[int, ..]*) – The row IDs that are to be deleted.

exit()
This signal is emitted when the client is exiting. Subscribers can use it as a chance to clean up and save any remaining data. It is emitted before the client is disconnected from the server. At this point the exit operation can not be cancelled.

Signal flags SIGNAL_ACTION | SIGNAL_RUN_LAST

exit-confirm()
This signal is emitted when the client has requested that the application exit. Subscribers to this signal can use it as a chance to display a warning dialog and cancel the operation.

Signal flags SIGNAL_ACTION | SIGNAL_RUN_LAST

message-delete(row_ids)
This signal is emitted when the user deletes a message entry. Subscribers to this signal can use it to conduct an action an entry is deleted.

Signal flags SIGNAL_ACTION | SIGNAL_RUN_LAST

Parameters **row_ids** (*[str, ..]*) – The row IDs that are to be deleted.

message-sent(target_uid, target_email)
This signal is emitted when the user sends a message. Subscribers to this signal can use it to conduct an action after the message is sent, and the information saved to the database.

Signal flags SIGNAL_RUN_FIRST

Parameters

- **target_uid** (*str*) – Message uid that was sent.
- **target_email** (*str*) – Email address associated with the sent message.

reload-css-style()
This signal is emitted to reload the style resources of the King Phisher client.

Signal flags SIGNAL_ACTION | SIGNAL_RUN_LAST

rpc-cache-clear()
This signal is emitted to clear the RPC objects cached information. Subsequent invocations of RPC cache enabled methods will return fresh information from the server.

Signal flags SIGNAL_ACTION | SIGNAL_RUN_LAST

server-connected()
This signal is emitted when the client has connected to the King Phisher server. The default handler sets the initial campaign optionally prompting the user to select one if one has not already been selected.

Signal flags SIGNAL_RUN_FIRST

server-disconnected()
This signal is emitted when the client has disconnected from the King Phisher server.

Signal flags SIGNAL_RUN_FIRST

sftp-client-start()
This signal is emitted when the client starts sftp client from within King Phisher. Subscribers can conduct an action prior to the default option being ran from the client configuration.

Signal flags SIGNAL_ACTION | SIGNAL_RUN_LAST

visit-delete(row_ids)
This signal is emitted when the user deletes a visit entry. Subscribers to this signal can use it to conduct an action an entry is deleted.

Signal flags SIGNAL_ACTION | SIGNAL_RUN_LAST

Parameters **row_ids** (*[str, ..]*) – The row IDs that are to be deleted.

unhandled-exception(exc_info, error_uid)
This signal is emitted when the application encounters an unhandled Python exception.

Signal flags SIGNAL_RUN_FIRST

Parameters

- **exc_info** (*tuple*) – A tuple of three objects corresponding to the return value of the sys.exc_info() function representing the exception that was raised.
- **error_uid** (uuid.UUID) – The unique identifier that has been assigned to this exception for tracking.

Mail Tab Signals

The following are the signals for the *MailSenderTab* object.

message-data-export(target_file)
This signal is emitted when the client is going to export the message configuration to a King Phisher Message (KPM) archive file.

Signal flags SIGNAL_ACTION | SIGNAL_RUN_LAST

Parameters **target_file** (*str*) – The path to write the archive file to.

Returns Whether or not the message archive was successfully imported.

Return type bool

message-data-import(target_file, dest_dir)
This signal is emitted when the client is going to import the message configuration from a King Phisher Message (KPM) archive file.

Signal flags SIGNAL_ACTION | SIGNAL_RUN_LAST

Parameters

- **target_file** (*str*) – The source archive file to import.
- **dest_dir** (*str*) – The destination directory to unpack the archive into.

Returns Whether or not the message archive was successfully imported.

Return type bool

send-finished()
This signal is emitted after all messages have been sent.

Signal flags SIGNAL_RUN_FIRST

send-precheck()
This signal is emitted when the user is about to start sending phishing messages. It is used to ensure that all settings are sufficient before proceeding. A handler can return False to indicate that a pre-check condition has failed and the operation should be aborted.

Signal flags SIGNAL_RUN_LAST

Returns Whether or not the handler's pre-check condition has passed.

Return type bool

send-target(target)
This signal is emitted when the target for a message has been loaded. Subscribers to this signal can use it as an opportunity to modify the target's attributes, including but not limited to the email address.

Signal flags SIGNAL_RUN_FIRST

Parameters **target** (*MessageTarget*) – The target for the message.

Server Event Signals

The following are the signals for the *ServerEventSubscriber* object. These events are published by the server forwarded to the client based on the active subscriptions. When an event is forwarded to a client the corresponding GObject signal is emitted for consumption by the client. See the section on *Published Events* for more details.

db-alert-subscriptions(event_type, objects)

Signal flags SIGNAL_RUN_FIRST

Parameters

- **event_type** (*str*) – The type of event, one of either deleted, inserted or updated.
- **objects** (*list*) – The objects from the server. The available attributes depend on the subscription.

db-campaigns(event_type, objects)

Signal flags SIGNAL_RUN_FIRST

Parameters

- **event_type** (*str*) – The type of event, one of either deleted, inserted or updated.
- **objects** (*list*) – The objects from the server. The available attributes depend on the subscription.

db-campaign-types(event_type, objects)

Signal flags SIGNAL_RUN_FIRST

Parameters

- **event_type** (*str*) – The type of event, one of either deleted, inserted or updated.
- **objects** (*list*) – The objects from the server. The available attributes depend on the subscription.

db-companies(event_type, objects)

Signal flags SIGNAL_RUN_FIRST

Parameters

- **event_type** (*str*) – The type of event, one of either deleted, inserted or updated.

- **objects** (*list*) – The objects from the server. The available attributes depend on the subscription.

db-company-departments(event_type, objects)

Signal flags SIGNAL_RUN_FIRST

Parameters

- **event_type** (*str*) – The type of event, one of either deleted, inserted or updated.
- **objects** (*list*) – The objects from the server. The available attributes depend on the subscription.

db-credentials(event_type, objects)

Signal flags SIGNAL_RUN_FIRST

Parameters

- **event_type** (*str*) – The type of event, one of either deleted, inserted or updated.
- **objects** (*list*) – The objects from the server. The available attributes depend on the subscription.

db-deaddrop-connections(event_type, objects)

Signal flags SIGNAL_RUN_FIRST

Parameters

- **event_type** (*str*) – The type of event, one of either deleted, inserted or updated.
- **objects** (*list*) – The objects from the server. The available attributes depend on the subscription.

db-deaddrop-deployments(event_type, objects)

Signal flags SIGNAL_RUN_FIRST

Parameters

- **event_type** (*str*) – The type of event, one of either deleted, inserted or updated.
- **objects** (*list*) – The objects from the server. The available attributes depend on the subscription.

db-industries(event_type, objects)

Signal flags SIGNAL_RUN_FIRST

Parameters

- **event_type** (*str*) – The type of event, one of either deleted, inserted or updated.
- **objects** (*list*) – The objects from the server. The available attributes depend on the subscription.

db-landing-pages(event_type, objects)

Signal flags SIGNAL_RUN_FIRST

Parameters

- **event_type** (*str*) – The type of event, one of either deleted, inserted or updated.
- **objects** (*list*) – The objects from the server. The available attributes depend on the subscription.

db-messages(event_type, objects)

Signal flags SIGNAL_RUN_FIRST

Parameters

- **event_type** (*str*) – The type of event, one of either deleted, inserted or updated.
- **objects** (*list*) – The objects from the server. The available attributes depend on the subscription.

db-users(event_type, objects)

Signal flags SIGNAL_RUN_FIRST

Parameters

- **event_type** (*str*) – The type of event, one of either deleted, inserted or updated.
- **objects** (*list*) – The objects from the server. The available attributes depend on the subscription.

db-visits(event_type, objects)

Signal flags SIGNAL_RUN_FIRST

Parameters

- **event_type** (*str*) – The type of event, one of either deleted, inserted or updated.
- **objects** (*list*) – The objects from the server. The available attributes depend on the subscription.

Keyboard Shortcuts

The following keyboard shortcuts are available for use within the client GUI.

Key Combination	Action Description
Ctrl + O	Open a campaign
Ctrl + Q	Exit the client
Ctrl + F1	Open an RPC terminal
Ctrl + F2	Open the SFTP client
Ctrl + Shift + F1	Clear the RPC cache
Ctrl + Shift + F2	Write the configuration to disk
Ctrl + Shift + F12	Reload the style css file

CHAPTER 3

The King Phisher Server

GraphQL

Overview

The RPC API provides a function for executing GraphQL queries against the server. The schema the server supports allows accessing the database models through the `db` type as well as some additional information such as the server plugins.

Note: For consistencies within the GraphQL API and with GraphQL best practices, it is important to note that names are `camelCase` and not `snake_case`.

Interface Extensions

The GraphQL schema supported by King Phisher implements the Relay connection interface allowing easier pagination using a cursor. As an extension to this interface, the King Phisher schema also includes a `total` attribute to the connection object. This attribute allows a query to access the number of nodes available for a specific connection.

Additional Database Model Attributes

Database objects which have an IP address string attribute associated with their model have an additional attribute containing the corresponding geo location information. This geo location attribute uses the same naming prefix, for example the geo location information for a `visitorIp` attribute can be accessed from the `visitorGeoloc` attribute.

Executing Raw Queries

Raw GraphQL queries can be executed using the `tools/database_console.py` utility. This console provides a `graphql_query` function which takes a query string parameter and optional query variables. This can be used for easily testing queries. It should be noted however that using this utility directly on the server does not restrict access to data as the RPC interface does.

Example Query

The following query is an example of retrieving the first 3 users from the users table. The query includes the necessary information to perform subsequent queries to iterate over all entries.

```
# GraphQL queries can have comments like this
query getFirstUser {
   # database objects are accessible under the 'db' type
   db {
      # retrieve the first 3 user objects
      users(first: 3) {
         # 'total' is an extension to the standard GraphQL relay interface
         total
         edges {
            # 'cursor' is a string used for iteration
            cursor
            node {
               # here the desired fields of the user object are specified
               id
               phoneNumber
            }
         }
         # request information regarding the chunk of users returned
         pageInfo {
            endCursor
            hasNextPage
         }
      }
   }
}
```

Published Events

Overview

Certain signals used by the server can be forwarded to clients via event subscriptions. In order to take advantage of this functionality the client opens a web socket to the server, and configures it's subscriptions using the available *Event API* functions. When a server signal is emitted the corresponding information is then forwarded to the subscribed clients over their open websocket.

Database Events

Database events can be subscribed to using the *event_id* of `db-TABLE_NAME`. Each of these events have the following sub-event types for each of the database operations.

- `deleted`

- `inserted`
- `updated`

These events are emitted by the respective `db_session_*` *Database Signals*. These signals are converted to events and organized by table (e.g. messages) instead of operation (e.g. inserted) because events are configured to send specific attributes. Not all attributes are available on all tables, however for one table the available attributes will always be available for all operations.

REST API

Overview

The King Phisher server provides an optional REST API *that is disabled by default*. It can be enabled by setting the server configuration value "rest_api.enabled" to true. An API token is required for all REST methods and must be present in the "token" parameter. If a static token is not specified in the server "rest_api.token" configuration, a new token will be randomly generated every time the server starts. The REST API methods are provided for access to convienence methods only. As such, campaign information can not be accessed via the REST API.

REST Methods

`GET /_/api/geoip/lookup`
Lookup an IP address in the GeoIP database.

Example request:

```
GET /_/api/geoip/lookup?token=SECRET_TOKEN&ip=4.2.2.2 HTTP/1.1
User-Agent: curl/7.40.0
Host: example.com
Accept: */*
```

Example response:

```
HTTP/1.0 200 OK
Server: Apache/2.4.12 (Unix)
Date: Thu, 04 Jun 2015 14:15:57 GMT
Content-Type: application/json
Content-Length: 204

{
  "result": {
   "city": null,
   "continent": "North America",
   "coordinates": [
     38.0,
     -97.0
   ],
   "country": "United States",
   "postal_code": null,
   "time_zone": null
  }
}
```

Query Parameters

- **token** – The server's REST API token.
- **ip** – The IP address to query geo location information for.

Status Codes

- 200 OK – The operation completed successfully.
- 401 Unauthorized – The REST API service is disabled or the token is invalid.
- 500 Internal Server Error – The operation encountered an exception.

GET /_/api/sms/send

Send an SMS message by emailing the carriers SMS gateway.

Example request:

```
GET /_/api/geoip/lookup?token=SECRET_TOKEN&message=hello+world!&phone_
↪number=1234567890&carrier=Sprint HTTP/1.1
User-Agent: curl/7.40.0
Host: example.com
Accept: */*
```

Example response:

```
HTTP/1.0 200 OK
Server: Apache/2.4.12 (Unix)
Date: Thu, 04 Jun 2015 14:30:40 GMT
Content-Type: application/json
Content-Length: 22

{
  "result": "sent"
}
```

Query Parameters

- **token** – The server's REST API token.
- **message** – The message to send.
- **phone_number** – The phone number to send the SMS to.
- **carrier** – The cellular carrier that the phone number belongs to.
- **from_address** – The optional address to display in the 'from' field of the SMS.

Status Codes

- 200 OK – The operation completed successfully.
- 401 Unauthorized – The REST API service is disabled or the token is invalid.
- 500 Internal Server Error – The operation encountered an exception.

RPC API

Overview

The RPC API is used by the King Phisher client to communicate with the server. It uses the RPC capabilities provided by the `AdvancedHTTPServer` module for the underlying communications. The RPC API provides a way for the

client to retrieve and set information regarding campaigns as well as the server's configuration. RPC requests must be authenticated and are only permitted from the loopback interface. The client is responsible for using SSH to set up a port forward for requests.

General API

graphql
query,, *query_vars=None*

Handler *rpc_graphql()*

login

Handler *rpc_login()*

logout

Handler *rpc_logout()*

ping

Handler *rpc_ping()*

plugins/list

Handler *rpc_plugins_list()*

shutdown

Handler *rpc_shutdown()*

version

Handler *rpc_version()*

Campaign API

campaign/alerts/is_subscribed
campaign_id

Handler *rpc_campaign_alerts_is_subscribed()*

campaign/alerts/subscribe
campaign_id

Handler *rpc_campaign_alerts_subscribe()*

campaign/alerts/unsubscribe
campaign_id

Handler *rpc_campaign_alerts_unsubscribe()*

campaign/landing_page/new
campaign_id,, *hostname,*, *page*

Handler *rpc_campaign_landing_page_new()*

campaign/message/new
campaign_id,, *email_id,*, *email_target,*, *company_name,*, *first_name,*, *last_name*

Handler *rpc_campaign_message_new()*

campaign/new
name,, *description=None*

Handler *rpc_campaign_new()*

campaign/stats
campaign_id

Handler *rpc_campaign_stats()*

Configuration API

config/get
option_name

Handler *rpc_config_get()*

config/set
options

Handler *rpc_config_set()*

Event API

events/is_subscribed
event_id,, *event_type*

Handler *rpc_events_is_subscribed()*

events/subscribe
event_id,, *event_types,*, *attributes*

Handler *rpc_events_subscribe()*

events/unsubscribe
event_id,, *event_types,*, *attributes*

Handler *rpc_events_unsubscribe()*

GeoIP API

geoip/lookup
ip,, *lang=None*

Handler *rpc_geoip_lookup()*

geoip/lookup/multi
ips,, *lang=None*

Handler *rpc_geoip_lookup_multi()*

Table API

db/table/count
table_name,, *query_filter=None*

Handler *rpc_database_count_rows()*

db/table/delete
table_name,, *row_id*

Handler *rpc_database_delete_row_by_id()*

db/table/delete/multi
table_name,, *row_ids*

Handler *rpc_database_delete_rows_by_id()*

db/table/get
table_name,, *row_id*

Handler *rpc_database_get_row_by_id()*

db/table/insert
table_name,, *keys,*, *values*

Handler *rpc_database_insert_row()*

db/table/set
table_name,, *row_id,*, *keys,*, *values*

Handler *rpc_database_set_row_value()*

db/table/view
table_name,, *page=0,*, *query_filter=None*

Handler *rpc_database_view_rows()*

Server Signals

Overview

Server signals are used by the server to dispatch events to subscribed handlers. This allows plugins to subscribe specific functions to be executed when a particular event occurs. These signals are defined in the *signals* module.

Database Signals

king_phisher.server.signals.**db_initialized**
Emitted after a connection has been made and the database has been fully initialized. At this point, it is safe to operate on the database.

Parameters **connection_url** (sqlalchemy.engine.url.URL) – The connection string for the database that has been initialized.

king_phisher.server.signals.**db_session_deleted**
Emitted after one or more rows have been deleted on a SQLAlchemy session. At this point, references are valid but objects can not be modified. See sqlalchemy.orm.events.SessionEvents.after_flush() for more details.

Parameters

- **table** (*str*) – The name of the table for which the target objects belong.
- **targets** (*tuple*) – The objects that have been deleted with the session.
- **session** (sqlalchemy.orm.session.Session) – The SQLAlchemy session with which the *targets* are associated.

king_phisher.server.signals.**db_session_inserted**
Emitted after one or more rows have been inserted in a SQLAlchemy session. At this point, references are valid but objects can not be modified. See sqlalchemy.orm.events.SessionEvents.after_flush() for more details.

Parameters

- **table** (*str*) – The name of the table for which the target objects belong.
- **targets** (*tuple*) – The objects that have been inserted with the session.
- **session** (sqlalchemy.orm.session.Session) – The SQLAlchemy session with which the *targets* are associated.

king_phisher.server.signals.**db_session_updated**
Emitted after one or more rows have been updated in a SQLAlchemy session. At this point, references are valid but objects can not be modified. See sqlalchemy.orm.events.SessionEvents.after_flush() for more details.

Parameters

- **table** (*str*) – The name of the table for which the target objects belong.
- **targets** (*tuple*) – The objects that have been updated with the session.
- **session** (sqlalchemy.orm.session.Session) – The SQLAlchemy session with which the *targets* are associated.

king_phisher.server.signals.**db_table_delete**
Emitted before a row inheriting from Base is deleted from the database table. To only subscribe to delete events for a specific table, specify the table's name as the *sender* parameter when calling blinker.base.Signal.connect(). See sqlalchemy.orm.events.MapperEvents.before_delete() for more details.

Parameters

- **table** (*str*) – The name of the table for which the target object belongs.
- **mapper** (sqlalchemy.orm.mapper.Mapper) – The Mapper object which is the target of the event.
- **connection** (sqlalchemy.engine.Connection) – The SQLAlchemy connection object which is being used to emit the SQL statements for the instance.
- **target** – The target object instance.

king_phisher.server.signals.**db_table_insert**
Emitted before a row inheriting from Base is inserted into the database table. To only subscribe to insert events for a specific table, specify the table's name as the *sender* parameter when calling blinker.base.Signal.connect(). See sqlalchemy.orm.events.MapperEvents.before_insert() for more details.

Parameters

- **table** (*str*) – The name of the table for which the target object belongs.
- **mapper** (sqlalchemy.orm.mapper.Mapper) – The Mapper object which is the target of the event.
- **connection** (sqlalchemy.engine.Connection) – The SQLAlchemy connection object which is being used to emit the SQL statements for the instance.
- **target** – The target object instance.

king_phisher.server.signals.**db_table_update**
Emitted before a row inheriting from Base is updated in the database table. To only subscribe to update events

for a specific table, specify the table's name as the *sender* parameter when calling `blinker.base.Signal.connect()`. See `sqlalchemy.orm.events.MapperEvents.before_update()` for more details.

Parameters

- **table** (*str*) – The name of the table for which the target object belongs.
- **mapper** (`sqlalchemy.orm.mapper.Mapper`) – The Mapper object which is the target of the event.
- **connection** (`sqlalchemy.engine.Connection`) – The SQLAlchemy connection object which is being used to emit the SQL statements for the instance.
- **target** – The target object instance.

Request Handler Signals

Signals which are emitted for events specific to individual HTTP requests. These signals use the respective instance of *KingPhisherRequestHandler* as the sender.

king_phisher.server.signals.**credentials_received**
Sent when a new pair of credentials have been submitted.

Parameters

- **request_handler** – The handler for the received request.
- **username** (*str*) – The username of the credentials that were submitted.
- **password** (*str*) – The password of the credentials that were submitted.

king_phisher.server.signals.**email_opened**
Sent when a request for the embedded image is received.

Parameters **request_handler** – The handler for the received request.

king_phisher.server.signals.**request_received**
Sent when a new HTTP request has been received and is about to be processed.

Parameters **request_handler** – The handler for the received request.

king_phisher.server.signals.**response_sent**
Sent after a response to an HTTP request has been sent to the client. At this point headers may be added to the response body.

Parameters

- **request_handler** – The handler for the received request.
- **code** (*int*) – The HTTP status code that was sent in the response.
- **message** (*str*) – The HTTP message that was sent in the response.

king_phisher.server.signals.**rpc_method_call**
Sent when a new RPC request has been received and it's corresponding method is about to be called.

Parameters

- **method** (*str*) – The RPC method which is about to be executed.
- **request_handler** – The handler for the received request.
- **args** (*tuple*) – The arguments that are to be passed to the method.
- **kwargs** (*dict*) – The key word arguments that are to be passed to the method.

king_phisher.server.signals.**rpc_method_called**
Sent after an RPC request has been received and it's corresponding method has been called.

Parameters

- **method** (*str*) – The RPC method which has been executed.
- **request_handler** – The handler for the received request.
- **args** (*tuple*) – The arguments that were passed to the method.
- **kwargs** (*dict*) – The key word arguments that were passed to the method.
- **retval** – The value returned from the RPC method invocation.

king_phisher.server.signals.**rpc_user_logged_in**
Sent when a new RPC user has successfully logged in and created a new authenticated session.

Parameters

- **request_handler** – The handler for the received request.
- **session** (*str*) – The session ID of the newly logged in user.
- **name** (*str*) – The username of the newly logged in user.

king_phisher.server.signals.**rpc_user_logged_out**
Sent when an authenticated RPC user has successfully logged out and terminated their authenticated session.

Parameters

- **request_handler** – The handler for the received request.
- **session** (*str*) – The session ID of the user who has logged out.
- **name** (*str*) – The username of the user who has logged out.

king_phisher.server.signals.**visit_received**
Sent when a new visit is received on a landing page. This is only emitted when a new visit entry is added to the database.

Parameters **request_handler** – The handler for the received request.

Server Signals

Signals which are emitted for a *KingPhisherServer* instance.

king_phisher.server.signals.**server_initialized**
Sent when a new instance of *KingPhisherServer* is initialized.

Parameters **server** – The newly initialized server instance.

CHAPTER 4

Plugins

Starting with version v1.3.0 King Phisher includes a plugin system. At this time only client plugins are supported with server side plugins slated for a future release. The common functionality for the two is provided by the *plugins* module and then extended by the respective implementation.

King Phisher supports loading plugins to allow the user to add additional features out side of what is supported by the main-stream application. These plugins are implemented as Python modules which define a *Plugin* class that is the respective plugins entry point.

Plugin Compatibility

Due to the way in which plugins are defined as classes with meta-data provided in their attributes, they need to be able to be imported regardless of compatibility restraints. The base *PluginBase* class offers a number of attributes which can be defined to indicate it's compatibility requirements.

Minimum King Phisher Version

A minimum required version of King Phisher can be specified in the *req_min_version* attribute. This should be used to indicate the version in which API changes were made that the plugin relies upon. The value of this attribute must be a string which can be parsed with Python's `distutils.version.StrictVersion` class for comparison.

The default class value is the first version of King Phisher which introduced the plugin API.

Required Python Packages

Sometimes modules may need additional Python packages and modules to be available in order to function properly. This can be problematic as the modules often need to be imported at the top level which normally would prevent the plugin from loading. In order to avoid this, plugin authors must wrap the import statement using Python's exception handling and define a variable to indicate whether or not the module is available.

This variable then needs to be added to the *req_packages* attribute. This attribute is a dictionary whose keys are the names of packages which are required with values of their availability. Using this method a plugin which requires the externally provided package "foo" can be loaded into King Phisher allowing it to correctly alert the user in the event that the "foo" package can not be loaded. It's highly recommended that the required packages be described in the plugins description.

Example

The following is a commented example of a basic client plugin with compatibility requirements.

```
import king_phisher.client.plugins as plugins
import king_phisher.client.gui_utilities as gui_utilities

try:
    import foobar
except ImportError:
    has_foobar = False  # catch the standard ImportError and set has_foobar to False
else:
    has_foobar = True   # no errors occurred so foobar was able to be imported

class Plugin(plugins.ClientPlugin):
    authors = ['Spencer McIntyre']
    title = 'Compatibility Demo'
    description = """
    A basic plugin which has compatibility requirements. It needs the 'foobar'
    Python package to be installed.
    """
    req_min_version = '1337.0'  # this is the required minimum version of King Phisher
    req_packages {
        'foobar': has_foobar    # whether or not foobar was able to be imported
    }
    # plugin method definitions continue
```

Client Plugins

Client plugins need to inherit from the *ClientPlugin* class which provides the basic outline. Client plugins have access to a dictionary for persistent configuration storage through the *config* attribute. In order for the plugin's meta-data to be available to the GUI, class attributes are used. This allows information such as the title, description, etc. to be accessed without initializing the class.

Plugin Manager

When the Plugin Manager window is loaded, all available plugins are loaded in order for their information to be retrieved from the class attributes. This is the effective equivalent of importing the module in Python. When the module is enabled, an instance of the Plugin class created allowing it to fulfill its intended purpose.

Reloading Plugins

Plugin modules and classes can be "reloaded" to allow changes made to the plugin on disk to take effect. This can be accomplished by right clicking the plugin and selecting the "Reload" option from the manager window. If an enabled plugin is reloaded, it will first be disabled before being re-enabled causing it to lose any state information it may have been storing.

Plugin Compatibility

Plugin modules that have requirements can have their compatibility checked by viewing the plugin information pane. This includes a simple yes or no regarding whether all of the plugin's requirements are met and the plugin is thus compatible. Additional information regarding the specific requirements a particular plugin has can be accessed by clicking the `Compatible` link which will show each of the requirements, their values and whether or not they are met. This allows users to easily determine why a particular plugin may not be compatible.

Plugin Options

Client plugins have special *ClientOption* classes available to them for specifying options that the user can set. The *king_phisher.client.plugins.ClientOptionMixin.__init__()* method offers additional parameters such as *display_name* to configure the information shown to the end user in the configuration widget.

The following are the different option classes available for client plugins:

- *ClientOptionBoolean*
- *ClientOptionEnum*
- *ClientOptionInteger*
- *ClientOptionPath*
- *ClientOptionPort*
- *ClientOptionString*

Example

The following is a commented example of a basic "Hello World" plugin.

```
import king_phisher.client.plugins as plugins
import king_phisher.client.gui_utilities as gui_utilities

# this is the main plugin class, it is necessary to inherit from plugins.ClientPlugin
class Plugin(plugins.ClientPlugin):
    authors = ['Spencer McIntyre']  # the plugins author
    title = 'Hello World!'          # the title of the plugin to be shown to users
    description = """
    A 'hello world' plugin to serve as a basic template and demonstration. This
    plugin will display a message box when King Phisher exits.
    """                             # a description of the plugin to be shown to users
    homepage = 'https://github.com/securestate/king-phisher-plugins'  # an optional␣
↪home page
    options = [  # specify options which can be configured through the GUI
        plugins.ClientOptionString(
            'name',                                 # the name of the option as it will␣
↪appear in the configuration
            'The name to which to say goodbye.',  # the description of the option as␣
↪shown to users
            default='Alice Liddle',                 # a default value for the option
            display_name='Your Name'                # a name of the option as shown to␣
↪users
        )
        plugins.ClientOptionBoolean(
            'validiction',
            'Whether or not this plugin say good bye.',
```

```
            default=True,
            display_name='Say Good Bye'
        ),
        plugins.ClientOptionInteger(
            'some_number',
            'An example number option.',
            default=1337,
            display_name='A Number'
        ),
        plugins.ClientOptionPort(
            'tcp_port',
            'The TCP port to connect to.',
            default=80,
            display_name='Connection Port'
        )
    ]
    req_min_version = '1.4.0'  # (optional) specify the required minimum version of
↪king phisher
    version = '1.0'            # (optional) specify this plugin's version
    # this is the primary plugin entry point which is executed when the plugin is
↪enabled
    def initialize(self):
        print('Hello World!')
        self.signal_connect('exit', self.signal_exit)
        # it is necessary to return True to indicate that the initialization was
↪successful
        # this allows the plugin to check its options and return false to indicate a
↪failure
        return True

    # this is a cleanup method to allow the plugin to close any open resources
    def finalize(self):
        print('Good Bye World!')

    # the plugin connects this handler to the applications 'exit' signal
    def signal_exit(self, app):
        # check the 'validiction' option in the configuration
        if not self.config['validiction']:
            return
        gui_utilities.show_dialog_info(
            "Good bye {0}!".format(self.config['name']),
            app.get_active_window()
        )
```

Server Plugins

Server plugins need to inherit from the *ServerPlugin* class which provides the basic outline. Server plugins have access to their respective configurations from the *config* attribute. This data is loaded from the server's configuration file and while it can be changed at runtime, the changes will not be kept after the server has stopped.

A plugin that needs to store data persistently can use the *storage* attribute which acts as a simple key value store and is backed by the database. Values stored in this must be able to be serialized making it impossible to directly store custom objects.

Server plugins can hook functionality by utilizing the *signals* module. This allows plugins to provide functionality for specific events.

Adding RPC Methods

Server plugins can provide new RPC methods that are available to client plugins and the client's RPC terminal. This allows server plugins to provide extended functionality for use by these other components.

Registering new RPC methods is as simple as calling the *register_rpc()* method. This function, like signal handlers, takes a method as an argument for use as a call back. This method is then called when the RPC function is invoked. The return value of this method is then returned to the caller of the RPC function. The method will automatically be passed the current *KingPhisherRequestHandler* instance as the first argument (after the standard "self" argument for class methods as applicable). Additional arguments after that accepted from the RPC invocation.

The following is an example of two custom RPC methods.

```
# ... other initialization code
class Plugin(plugins.ServerPlugin):
    # ... other initialization code
    def initialize(self):
         self.register_rpc('add', self.rpc_add)
         self.register_rpc('greet', self.rpc_greet)
         return True

    # this example takes two arguments to be invoked and returns their sum
    # >>> rpc('plugins/example/add', 1, 2)
    # 3
    def rpc_add(self, handler, number_1, number_2):
        return number_1 + number_2

    # this example takes no arguments but accesses the rpc_session to
    # retrieve the current user name
    # >>> rpc('plugins/example/greet')
    # 'Hello steiner'
    def rpc_greet(self, handler):
        rpc_session = handler.rpc_session
        return 'Hello ' + rpc_session.user
```

Example

The following is a commented example of a basic "Hello World" plugin.

```
import king_phisher.plugins as plugin_opts
import king_phisher.server.plugins as plugins
import king_phisher.server.signals as signals

# this is the main plugin class, it is necessary to inherit from plugins.ServerPlugin
class Plugin(plugins.ServerPlugin):
    authors = ['Spencer McIntyre']  # the plugins author
    title = 'Hello World!'
    description = """
    A 'hello world' plugin to serve as a basic template and demonstration.
    """
    homepage = 'https://github.com/securestate/king-phisher-plugins'
    options = [  # specify options which need to be set through the configuration file
        plugin_opts.OptionString(
            'name',               # the options name
            'the name to greet',  # a basic description of the option
            default=None          # a default value can be specified to
```

```
        )
    ]
    req_min_version = '1.4.0'      # (optional) specify the required minimum version
↪of king phisher
    version = '1.0'                # (optional) specify this plugin's version
    def initialize(self):
        self.logger.warning('hello ' + self.config['name'] + '!')
        # connect to a signal via it's object in the signals module
        signals.server_initialized.connect(self.on_server_initialized)
        return True

    def on_server_initialized(self, server):
        self.logger.warning('the server has been initialized')
```

CHAPTER 5

Development References

Style Guide

It's important for a project to have a standardized style for it's code. The King Phisher project, being a Python project follows the PEP-8 style guide. With the following notable exceptions:

- Do use hard tabs instead of spaces.
- Do not use more than one consecutive blank line, ever.
- Do limit lines of code to 120 characters long instead of 79.
  - Do limit documentation lines to 80 characters long.
- Do use single quotes for strings with the exception of template strings (such as those used by `str.format`) and documentation strings which should use triple double-quotes.

Multi Line Indentation

Use hanging indentation for parenthesized statements. This is to say, the last non-whitespace character of the line should be the opening parenthesis with each subsequent line being indented until the closing parenthesis. Furthermore, in the case that this style is used, each expression should be on a separate line.

Example:

```
# good (standard one-line invocation)
this_function(takes_two, different_arguments)

# good (multi-line invocation)
this_other_function_has_a_longer_name(
    and_also_takes_two,
    different_arguments
)

# bad
```

```
this_other_function_has_a_longer_name(and_one_argument_up_here,
    and_another_down_here
)
```

This same style is applied to multi-line list, tuple and dictionary definitions with the bracket, or curly-brace taking the place of the opening and closing parenthesis as appropriate.

English Verbiage

Use full, complete and grammatically correct sentences for all documentation purposes. This includes class, function, attribute, and parameter documentation. Additionally, proper sentences should be used for any messages that are displayed to end users with the notable exception of log messages. Log messages are to be entirely lowercase with the exception of acronyms which are currently inconsistently cased. Either all capital letters or all lower case letters are acceptable for acronyms within log messages.

Release Steps

This document contains the steps that are followed for each point version release of King Phisher.

Pre Release Steps

1. Test and fix any issues with the Windows MSI build
2. Ensure unit tests pass with Python 2.7 & Python 3.5
3. Remove the version label
4. Create the final Windows MSI build
5. Update the change log

Release Steps

1. Create a final signed commit on the dev branch and push it to GitHub
2. Merge dev into master and push master to GitHub
3. Create and push a signed tag of the release commit
4. Create a new release on GitHub
   (a) Upload the final Windows build
   (b) Insert the changes from the change log
   (c) Insert the MD5 and SHA1 hashes of the Windows build
5. Update the Docker build
6. Publicize the release

Post Release Steps

1. Increment the version number on the dev branch and re-set the version label
2. Update Python packages list for pip in requirements.txt with piprot

```
python3 -m pip install -U piprot
sed -e 's/>=/==/g' requirements.txt | \
piprot -x - | \
awk '/# Latest/ {print substr($1, 0, index($1, "==") + 1) $4 }'
```

Windows Build

Each release of King Phisher includes an MSI build of the client for easy use on Windows systems. Creating this build is one of the last steps prior to creating a new version release. The build is created using the Python cx_Freeze package.

Before the build can be created the PyGObject for Windows package must be installed. While installing this package, it prompts for which GNOME libraries are to be included. When the prompt appears the following packages should be selected.

- Base packages
- ATK
- GConf
- GDK-Pixbuf
- GTK+
- GTKSourceView
- GXML
- Pango
- Soup
- WebkitGTK

Once all packages have been installed and the King Phisher client is running with Python, the "tools/build_msi.bat" script can be executed to create the build. The build process will take a few minutes, but once completed an MSI installer file will be created in a new "dist" directory in the projects root folder.

Version Information

After building the MSI file you will need to add custom properties. By right clicking on the MSI file, select properties, and then the custom tab you can add custom fields. You will need to add the Python Version, and PyGI-AIO version utilized in making the build as text entries. Below is the name fields and example values.

Name	Example Value
Python Version	2.7.11
PyGI-AIO Version	3.14.0 rev22

CHAPTER 6

Environment Variables

The following environment variables can be set to change normal operation. None of them are required to be set under normal circumstances.

Variable Name	Variable Description
KING_PHISHER_DATA_PATH	Paths to search for data files
KING_PHISHER_GLADE_FILE	Name of the client Glade UI data file
KING_PHISHER_TEST_GEOIP_DB	The GeoIP database used for unit tests
KING_PHISHER_TEST_OFFLINE	Skip unit tests which require a network connection

CHAPTER 7

Change Log

This document contains notes on the major changes for each version of King Phisher.

Version 1.x.x

Version 1.8.0

In Progress

- Install script now supports Red Hat Server 7
- Support the client on OS X by using Docker
- Support for issuing certificates with acme while the server is running
- Add a wrapping tool for certbot to make the process easier

Version 1.7.1

Released v1.7.1 on April 14th, 2017

- Bug fix in the Windows build for HTTPS connections from the requests package

Version 1.7.0

Released v1.7.0 on April 4th, 2017

- Better error messages for malformed server configuration files
- Support for sending to targets via To / CC / BCC fields
- New features for client and server plugins
- Add comparison of "trained" statistics to the campaign comparison

- Support for including and importing Jinja templates from relative paths
- Support for including custom HTTP headers in server responses
- New feature to import Campaigns from XML files
- Support for emails address with longer top level domain names

Version 1.6.0

Released v1.6.0 on January 31st, 2017

- Support negotiating STARTTLS with SMTP servers that support it
- Support for real time event publishing to the client
- Support for a new GraphQL API for more efficient data queries
- More flexibility in configuring server logging
- Add persistent storage for server plugin data
- Add a Jinja function to check if a password is complex
- Add client `message-data-export` and `message-data-import` signals
- King Phisher now starts with Python3 by default
- `tools/install.sh` now creates a backup of server_config.yml when present
- Minor bug fixes
  - Minor CSS fixes
  - Special characters now display in the UI correctly

Version 1.5.2

Released v1.5.2 on December 23rd, 2016

- Minor bug fixes
  - Use Default SMS sender to fix SMS subscription with T-Mobile
  - Upgrade AdvancedHTTPServer to v2.0.6 to fix select polling
  - Corrected issue when attachment file is inaccessible
  - Fixed issue when message file directory is gone
  - Fixed server side encoding error with basic auth
  - Fixed TypeError handling while rendering templates
  - Fixed a unicode bug when processing targets csv
  - Fixed install.sh script for CentOS7 and python3
  - Fixed show exception dialog with Glib idle_add
  - Fixed a logic bug causing premature SMTP reconnects
  - Fixed Webkit-1 load_string Null error

Version 1.5.1

Released v1.5.1 on October 3rd, 2016

- Automated installation script improvements
  - Backup an existing server configuration file
  - Log warnings when the PostgreSQL user exists
- Improve the Metasploit plugin for session notifications via SMS
- Support exporting credentials for use with Metasploit's `USERPASS_FILE` option

Version 1.5.0

Released v1.5.0 on September 22nd, 2016

- Added an SPF button to the client for on demand SPF record checking
- Fixed missing packages in the Windows build for timezone data
- Transitioned to the dnspython package for Python 2.x and 3.x

Version 1.4.0

Released v1.4.0 on August 5th, 2016

- Added additional Jinja variables for server pages
- Upgraded to AdvancedHTTPServer version 2
  - Added support for binding to multiple interfaces
  - Added support for multiple SSL hostnames via SNI
- Support for plugins in the server application
- Added server signals for event subscriptions in plugins
- Updated the style for GTK 3.20
- Start to warn users about the impending Python 2.7 deprecation
- Change to installing for Python 3
- Added an uninstallation script

Version 1.3.0

Released v1.3.0 on May 17th, 2016

- Added automatic setup of PostgreSQL database for the server
- Server bug fixes when running on non-standard HTTP ports
- Added completion to the messaged editor
- Support for plugins in the client application
- Added a client plugin to automatically check for updates
- Added a client plugin to generate anonmous statistics

- Added debug logging of parameters for key RPC methods
- Lots of Python 3.x compatiblity fixes

Version 1.2.0

Released v1.2.0 on March 18th, 2016

- SSH host key validation
- Install script command line flags
- Support for authenticating to SMTP servers
- Style and compatibility changes for Kali

Version 1.1.0

Released v1.1.0 on December 30th, 2015

- Added an option to send a message to a single target
- Support for sending calendar invite messages
- Added PostgreSQL setup to the installer
- Support for exporting to Excel
- Added a Jupyter notebook for interactive data analysis
- Added additional campaign filtering options
- Support for removal of metadata from Microsoft Office 2007+ documents

Version 1.0.0

Released v1.0.0 on October 15th, 2015

- Moved templates to a dedicated separate repository
- Added a custom theme for the client
- Added support for two factor authentication with TOTP
- Support for specifying an img style attribute for inline images in messages

Version 0.x.x

Version 0.3.0

Released v0.3.0 on August 21st, 2015

- Added a new campaign creation assistant
- Support for expiring campaigns at a specified time
- Track more details when messages are opened such as the IP address and User Agent
- Support for tagging campaign types

- Support for organizing campaigns by companies
- Support for storing email recipients department name
- Support for collecting credentials via Basic Auth

Version 0.2.1

Released v0.2.1 on July 14th, 2015

- Added syntax highlighting to the message edit tab
- Technical documentation improvements, including documenting the REST API
- Support reloading message templates when they change from an external editor
- Support for pulling the client IP from a cookie set by an upstream proxy
- Support for embedding training videos from YouTube
- Added a Metasploit plugin for using the REST API to send SMS messages
- Support for exporting visit information to GeoJSON

Version 0.2.0

Released v0.2.0 on April 28th, 2015

- Added additional graphs including maps when basemap is available
- Added geolocation support
- Made dashboard layout configurable
- Support for cloning web pages
- Support for installing on Fedora
- Support for running the server with Docker

Version 0.1.7

Released v0.1.7 on February 19th, 2015

- Added make_csrf_page function
- Added server support for SSL
- Support verifying the server configuration file
- Added a desktop file and icon for the client GUI
- Added support for operating on multiple rows in the client's campaign tables
- Support starting an external SFTP application from the client
- Tweaked miscellaneous features to scale for larger campaigns (35k+ messages)
- Updated AdvancedHTTPServer to version 0.4.2 which supports Python 3
- Added integration for checking Sender Policy Framework (SPF) records

Version 0.1.6

Released v0.1.6 on November 3rd, 2014

- Migrated to SQLAlchemy backend (SQLite will no longer be supported for database upgrades)
- Added additional documentation to the wiki
- Enhanced error handling and UI documentation for a better user experience
- Support for quickly adding common dates and times in the message editor

Version 0.1.5

Released v0.1.5 on September 29th, 2014

- Added support for inline images in emails
- Import and export support for message configurations
- Highlight the current campaign in the selection dialog

Version 0.1.4

Released v0.1.4 on September 4th, 2014

- Full API documentation
- Install script for Kali & Ubuntu
- Lots of bug fixes

Version 0.1.3

Released v0.1.3 on June 4th, 2014

- Jinja2 templates for both the client and server
- API version checking to warn when the client and server versions are incompatible

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

/_

GET /_/api/geoip/lookup, 109
GET /_/api/sms/send, 110

Python Module Index

a

archive, 69
assistants.campaign, 3

c

client.application, 20
client.client_rpc, 22
client.export, 24
client.graphs, 26
client.gui_utilities, 30
client.mailer, 36
client.plugins, 40
client.server_events, 44
client.web_cloner, 46
client.windows.campaign_import, 18
client.windows.compare_campaigns, 18
client.windows.main, 19
client.windows.plugin_manager, 19
client.windows.rpc_terminal, 20
color, 70
constants, 72

d

database.manager, 47
database.models, 49
database.storage, 50
dialogs.about, 4
dialogs.campaign_selection, 4
dialogs.clone_page, 4
dialogs.company_editor, 4
dialogs.configuration, 4
dialogs.entry, 5
dialogs.exception, 5
dialogs.login, 6
dialogs.ssh_host_key, 7
dialogs.tag_editor, 7

e

errors, 73

f

find, 74

g

geoip, 75

i

ics, 76
ipaddress, 78

p

plugins, 81

s

scrubber, 85
serializers, 85
server.aaa, 51
server.build, 53
server.graphql, 54
server.pages, 55
server.plugins, 55
server.rest_api, 56
server.server, 57
server.server_rpc, 58
server.signals, 64
server.web_sockets, 67
sms, 87
smtp_server, 88
spf, 88
ssh_forward, 91

t

tabs.campaign, 8
tabs.mail, 10
templates, 91
testing, 92

u

ua_parser, 93

utilities, 94

V

version, 97

W

widget.completion_providers, 17
widget.extras, 13
widget.managers, 15
widget.resources, 17

X

xor, 98

Index

Symbols

__geo_interface__ (king_phisher.geoip.GeoLocation attribute), 76
__init__() (king_phisher.archive.ArchiveFile method), 69
__init__() (king_phisher.client.assistants.campaign.CampaignAssistant method), 3
__init__() (king_phisher.client.dialogs.exception.ExceptionDialog method), 6
__init__() (king_phisher.client.dialogs.ssh_host_key.BaseHostKeyDialog method), 7
__init__() (king_phisher.client.dialogs.ssh_host_key.MissingHostKeyPolicy method), 7
__init__() (king_phisher.client.graphs.GraphBase method), 27
__init__() (king_phisher.client.gui_utilities.GladeDependencies method), 34
__init__() (king_phisher.client.gui_utilities.GladeGObject method), 35
__init__() (king_phisher.client.gui_utilities.GladeProxy method), 36
__init__() (king_phisher.client.gui_utilities.GladeProxyDestination method), 36
__init__() (king_phisher.client.mailer.MailSenderThread method), 37
__init__() (king_phisher.client.mailer.TopMIMEMultipart method), 40
__init__() (king_phisher.client.plugins.ClientOptionBoolean method), 40
__init__() (king_phisher.client.plugins.ClientOptionEnum method), 41
__init__() (king_phisher.client.plugins.ClientOptionInteger method), 41
__init__() (king_phisher.client.plugins.ClientOptionMixin method), 41
__init__() (king_phisher.client.plugins.ClientOptionPath method), 42
__init__() (king_phisher.client.plugins.ClientOptionPort method), 42
__init__() (king_phisher.client.plugins.ClientOptionString method), 42
__init__() (king_phisher.client.server_events.ServerEventSubscriber method), 45
__init__() (king_phisher.client.tabs.campaign.CampaignViewTab method), 10
__init__() (king_phisher.client.tabs.mail.MailSenderPreviewTab method), 11
__init__() (king_phisher.client.tabs.mail.MailSenderTab method), 13
__init__() (king_phisher.client.web_cloner.WebPageCloner method), 46
__init__() (king_phisher.client.widget.extras.FileChooserDialog method), 14
__init__() (king_phisher.client.widget.extras.WebKitHTMLView method), 14
__init__() (king_phisher.client.widget.managers.RadioButtonGroupManage method), 15
__init__() (king_phisher.client.widget.managers.TreeViewManager method), 15
__init__() (king_phisher.client.windows.main.MainAppWindow method), 19
__init__() (king_phisher.client.windows.rpc_terminal.RPCTerminal method), 20
__init__() (king_phisher.errors.KingPhisherAbortRequestError method), 73
__init__() (king_phisher.geoip.GeoLocation method), 76
__init__() (king_phisher.ics.Calendar method), 77
__init__() (king_phisher.ics.Timezone method), 78
__init__() (king_phisher.plugins.OptionBase method), 81
__init__() (king_phisher.plugins.OptionBoolean method), 82
__init__() (king_phisher.plugins.OptionEnum method), 82
__init__() (king_phisher.plugins.OptionInteger method), 82
__init__() (king_phisher.plugins.OptionString method), 82
__init__() (king_phisher.plugins.PluginManagerBase method), 84
__init__() (king_phisher.server.aaa.AuthenticatedSession

method), 51
__init__() (king_phisher.server.aaa.AuthenticatedSessionManager method), 51
__init__() (king_phisher.server.aaa.CachedPassword method), 52
__init__() (king_phisher.server.aaa.ForkedAuthenticator method), 52
__init__() (king_phisher.server.database.storage.KeyValueStorage method), 50
__init__() (king_phisher.server.server.KingPhisherServer method), 58
__init__() (king_phisher.server.web_sockets.EventSocket method), 67
__init__() (king_phisher.server.web_sockets.WebSocketsManager method), 68
__init__() (king_phisher.smtp_server.BaseSMTPServer method), 88
__init__() (king_phisher.spf.SenderPolicyFramework method), 89
__init__() (king_phisher.ssh_forward.SSHTCPForwarder method), 91
__init__() (king_phisher.templates.TemplateEnvironmentBase method), 92

A

AboutDialog (class in king_phisher.client.dialogs.about), 4
add() (king_phisher.server.web_sockets.WebSocketsManager method), 68
add_attendee() (king_phisher.ics.Calendar method), 77
add_data() (king_phisher.archive.ArchiveFile method), 69
add_file() (king_phisher.archive.ArchiveFile method), 69
add_menu_item() (king_phisher.client.plugins.ClientPlugin method), 43
add_reference() (king_phisher.client.application.KingPhisherClient method), 21
add_submenu() (king_phisher.client.plugins.ClientPlugin method), 43
adjust_path() (king_phisher.server.server.KingPhisherRequestHandler method), 57
ANDROID (king_phisher.constants.OSFamily attribute), 73
application (king_phisher.client.gui_utilities.GladeGObject attribute), 35
application (king_phisher.client.plugins.ClientPlugin attribute), 43
archive (module), 69
ArchiveFile (class in king_phisher.archive), 69
argp_add_args() (in module king_phisher.utilities), 94
args (king_phisher.client.gui_utilities.GladeProxyDestination attribute), 36
assert_arg_type() (in module king_phisher.utilities), 94
assert_session_has_permissions() (king_phisher.server.database.models.BaseRowCls method), 50
assertHTTPStatus() (king_phisher.testing.KingPhisherServerTestCase method), 93
assertRPCPermissionDenied() (king_phisher.testing.KingPhisherServerTestCase method), 93
assistants.campaign (module), 3
attachment_images (king_phisher.templates.MessageTemplateEnvironment attribute), 92
authenticate() (king_phisher.server.aaa.ForkedAuthenticator method), 53
AuthenticatedSession (class in king_phisher.server.aaa), 51
AuthenticatedSessionManager (class in king_phisher.server.aaa), 51
AuthorizationMiddleware (class in king_phisher.server.graphql), 54
authors (king_phisher.plugins.PluginBase attribute), 83
available (king_phisher.plugins.PluginManagerBase attribute), 84

B

BaseHostKeyDialog (class in king_phisher.client.dialogs.ssh_host_key), 7
BaseRowCls (class in king_phisher.server.database.models), 50
BaseSMTPServer (class in king_phisher.smtp_server), 88
BLACKBERRY (king_phisher.constants.OSFamily attribute), 73
build_prompt() (king_phisher.client.dialogs.entry.TextEntryDialog class method), 5

C

ClientApplication
cache (king_phisher.server.aaa.ForkedAuthenticator attribute), 53
cache_timeout (king_phisher.server.aaa.ForkedAuthenticator attribute), 53
CachedPassword (class in king_phisher.server.aaa), 52
Calendar (class in king_phisher.ics), 77
campaign_id (king_phisher.server.server.KingPhisherRequestHandler attribute), 57
campaign_name (king_phisher.client.assistants.campaign.CampaignAssistant attribute), 3
campaign_rename() (king_phisher.client.application.KingPhisherClientApplication method), 21
campaign_to_xml() (in module king_phisher.client.export), 24
campaign_visits_to_geojson() (in module king_phisher.client.export), 25
CampaignAssistant (class in king_phisher.client.assistants.campaign),

3
CampaignCompGraph (class in king_phisher.client.graphs), 30
CampaignCompWindow (class in king_phisher.client.windows.compare_campaigns), 19
CampaignGraph (class in king_phisher.client.graphs), 28
CampaignGraphMessageResults (class in king_phisher.client.graphs), 28
CampaignGraphOverview (class in king_phisher.client.graphs), 28
CampaignGraphPasswordComplexityPie (class in king_phisher.client.graphs), 28
CampaignGraphVisitorInfo (class in king_phisher.client.graphs), 29
CampaignGraphVisitorInfoPie (class in king_phisher.client.graphs), 29
CampaignGraphVisitsMap (class in king_phisher.client.graphs), 29
CampaignGraphVisitsMapUSA (in module king_phisher.client.graphs), 29
CampaignGraphVisitsMapWorld (in module king_phisher.client.graphs), 29
CampaignGraphVisitsTimeline (class in king_phisher.client.graphs), 29
CampaignSelectionDialog (class in king_phisher.client.dialogs.campaign_selection), 4
CampaignViewCredentialsTab (class in king_phisher.client.tabs.campaign), 8
CampaignViewDashboardTab (class in king_phisher.client.tabs.campaign), 8
CampaignViewDeaddropTab (class in king_phisher.client.tabs.campaign), 8
CampaignViewGenericTab (class in king_phisher.client.tabs.campaign), 8
CampaignViewGenericTableTab (class in king_phisher.client.tabs.campaign), 9
CampaignViewMessagesTab (class in king_phisher.client.tabs.campaign), 10
CampaignViewTab (class in king_phisher.client.tabs.campaign), 10
CampaignViewVisitsTab (class in king_phisher.client.tabs.campaign), 10
CARRIERS (in module king_phisher.sms), 87
cb_delete (king_phisher.client.widget.managers.TreeViewManager attribute), 16
cb_refresh (king_phisher.client.widget.managers.TreeViewManager attribute), 16
CellRendererBytes (class in king_phisher.client.widget.extras), 13
check_host() (in module king_phisher.spf), 89
check_host() (king_phisher.spf.SenderPolicyFramework method), 89
child_pid (king_phisher.server.aaa.ForkedAuthenticator attribute), 53
child_routine() (king_phisher.server.aaa.ForkedAuthenticator method), 53
children (king_phisher.client.gui_utilities.GladeDependencies attribute), 34
children (king_phisher.client.gui_utilities.GladeProxy attribute), 36
children (king_phisher.client.widget.resources.CompanyEditorGrid attribute), 17
city (king_phisher.geoip.GeoLocation attribute), 76
clean() (king_phisher.server.aaa.AuthenticatedSessionManager method), 51
clear_database() (in module king_phisher.server.database.manager), 47
client.application (module), 20
client.client_rpc (module), 22
client.export (module), 24
client.graphs (module), 26
client.gui_utilities (module), 30
client.mailer (module), 36
client.plugins (module), 40
client.server_events (module), 44
client.web_cloner (module), 46
client.windows.campaign_import (module), 18
client.windows.compare_campaigns (module), 18
client.windows.main (module), 19
client.windows.plugin_manager (module), 19
client.windows.rpc_terminal (module), 20
ClientOptionBoolean (class in king_phisher.client.plugins), 40
ClientOptionEnum (class in king_phisher.client.plugins), 41
ClientOptionInteger (class in king_phisher.client.plugins), 41
ClientOptionMixin (class in king_phisher.client.plugins), 41
ClientOptionPath (class in king_phisher.client.plugins), 42
ClientOptionPort (class in king_phisher.client.plugins), 42
ClientOptionString (class in king_phisher.client.plugins), 42
ClientPlugin (class in king_phisher.client.plugins), 43
ClientPluginMailerAttachment (class in king_phisher.client.plugins), 44
ClientPluginManager (class in king_phisher.client.plugins), 44
cloned_resources (king_phisher.client.web_cloner.WebPageCloner attribute), 46
ClonedResourceDetails (class in king_phisher.client.web_cloner), 46
ClonePageDialog (class in king_phisher.client.dialogs.clone_page),

4
close() (king_phisher.archive.ArchiveFile method), 70
color (module), 70
color_with_creds (king_phisher.client.graphs.CampaignGraphVisitsMap attribute), 29
color_without_creds (king_phisher.client.graphs.CampaignGraphVisitsMap attribute), 29
ColoredLogFormatter (class in king_phisher.color), 72
column_titles (king_phisher.client.widget.managers.TreeViewManager attribute), 16
column_views (king_phisher.client.widget.managers.TreeViewManager attribute), 16
commit() (king_phisher.client.client_rpc.RemoteRow method), 24
CompanyEditorDialog (class in king_phisher.client.dialogs.company_editor), 4
CompanyEditorGrid (class in king_phisher.client.widget.resources), 17
compatibility (king_phisher.plugins.PluginBaseMeta attribute), 83
config (king_phisher.client.application.KingPhisherClientApplication attribute), 21
config (king_phisher.client.graphs.GraphBase attribute), 27
config (king_phisher.client.gui_utilities.GladeGObject attribute), 35
config (king_phisher.client.plugins.ClientPlugin attribute), 43
config (king_phisher.client.windows.main.MainAppWindow attribute), 19
config (king_phisher.plugins.PluginBase attribute), 83
config (king_phisher.server.plugins.ServerPlugin attribute), 55
config (king_phisher.server.server.KingPhisherRequestHandler attribute), 57
config (king_phisher.server.server.KingPhisherServer attribute), 58
config_file (king_phisher.client.application.KingPhisherClientApplication attribute), 21
config_prefix (king_phisher.client.gui_utilities.GladeGObject attribute), 35
CONFIG_READABLE (in module king_phisher.server.server_rpc), 58
CONFIG_WRITEABLE (in module king_phisher.server.server_rpc), 58
ConfigurationDialog (class in king_phisher.client.dialogs.configuration), 4
configure_stream_logger() (in module king_phisher.utilities), 95
ConnectionErrorReason (class in king_phisher.constants), 72
ConstantGroup (class in king_phisher.constants), 72
constants (module), 72
continent (king_phisher.geoip.GeoLocation attribute), 76
convert_hex_to_tuple() (in module king_phisher.color), 71
convert_tuple_to_hex() (in module king_phisher.color), 71
create_table() (in module king_phisher.client.export), 25
Coordinates (class in king_phisher.geoip), 75
coordinates (king_phisher.geoip.GeoLocation attribute), 76
copy_resource_data() (king_phisher.client.web_cloner.WebPageCloner method), 46
count_messages() (king_phisher.client.mailer.MailSenderThread method), 37
country (king_phisher.geoip.GeoLocation attribute), 76
create_calendar_invite() (king_phisher.client.mailer.MailSenderThread method), 38
create_email() (king_phisher.client.mailer.MailSenderThread method), 38
created (king_phisher.server.aaa.AuthenticatedSession attribute), 51
credentials_received (in module king_phisher.server.signals), 64
current_timestamp() (in module king_phisher.server.database.models), 49
CustomCompletionProviderBase (in module king_phisher.client.widget.completion_providers), 17

D

data_directory() (in module king_phisher.find), 74
DATA_DIRECTORY_NAME (in module king_phisher.find), 74
data_file() (in module king_phisher.find), 74
data_path_append() (in module king_phisher.find), 74
database.manager (module), 47
database.models (module), 49
database.storage (module), 50
database_table_objects (in module king_phisher.server.database.models), 49
database_tables (in module king_phisher.server.database.models), 49
datetime_local_to_utc() (in module king_phisher.utilities), 95
datetime_utc_to_local() (in module king_phisher.utilities), 95
DAY_ABBREVIATIONS (in module king_phisher.ics), 76
DB_DOWNLOAD_URL (in module king_phisher.geoip), 75
db_initialized (in module king_phisher.server.signals), 64
DB_RESULT_FIELDS (in module king_phisher.geoip), 75

db_session_deleted (in module king_phisher.server.signals), 64
db_session_inserted (in module king_phisher.server.signals), 64
db_session_updated (in module king_phisher.server.signals), 65
db_table_delete (in module king_phisher.server.signals), 65
db_table_insert (in module king_phisher.server.signals), 65
db_table_update (in module king_phisher.server.signals), 65
DEFAULT_FROM_ADDRESS (in module king_phisher.sms), 87
DEFAULT_LOG_LEVEL (in module king_phisher.constants), 72
default_response (king_phisher.client.dialogs.ssh_host_key.BaseHostKeyDialog attribute), 7
department (king_phisher.client.mailer.MessageTarget attribute), 40
dependencies (king_phisher.client.gui_utilities.GladeGObject attribute), 35
description (king_phisher.plugins.PluginBase attribute), 83
destination (king_phisher.client.gui_utilities.GladeProxy attribute), 36
destroy() (king_phisher.client.gui_utilities.GladeGObject method), 35
dialogs.about (module), 4
dialogs.campaign_selection (module), 4
dialogs.clone_page (module), 4
dialogs.company_editor (module), 4
dialogs.configuration (module), 4
dialogs.entry (module), 5
dialogs.exception (module), 5
dialogs.login (module), 6
dialogs.ssh_host_key (module), 7
dialogs.tag_editor (module), 7
disable() (king_phisher.plugins.PluginManagerBase method), 84
DISABLED (in module king_phisher.constants), 72
dispatch() (king_phisher.server.web_sockets.WebSocketsManager method), 68
distutils_version (in module king_phisher.version), 97
do_campaign_delete() (king_phisher.client.application.KingPhisherClientApplication method), 21
do_config_load() (king_phisher.client.application.KingPhisherClientApplication method), 21
do_open_remote_uri() (king_phisher.client.widget.extras.WebKitHTMLView method), 14
do_server_disconnected() (king_phisher.client.application.KingPhisherClientApplication method), 21
do_sftp_client_start() (king_phisher.client.application.KingPhisherClientApplication method), 21
download_geolite2_city_db() (in module king_phisher.geoip), 75
draw_states (king_phisher.client.graphs.CampaignGraphVisitsMap attribute), 29
dst_end (king_phisher.ics.TimezoneOffsetDetails attribute), 78
dst_start (king_phisher.ics.TimezoneOffsetDetails attribute), 78
dump() (king_phisher.serializers.Serializer class method), 87
dumps() (king_phisher.serializers.JSON class method), 86
dumps() (king_phisher.serializers.MsgPack class method), 86
DurationAllDay (class in king_phisher.ics), 77

E

email_address (king_phisher.client.mailer.MessageTarget attribute), 40
email_opened (in module king_phisher.server.signals), 66
embed_youtube_video() (in module king_phisher.server.pages), 55
enable() (king_phisher.plugins.PluginManagerBase method), 84
enabled_plugins (king_phisher.plugins.PluginManagerBase attribute), 84
encoding (king_phisher.serializers.Serializer attribute), 87
ENV_VAR (in module king_phisher.find), 74
ERROR_AUTHENTICATION_FAILED (king_phisher.constants.ConnectionErrorReason attribute), 72
ERROR_CONNECTION (king_phisher.constants.ConnectionErrorReason attribute), 72
ERROR_INCOMPATIBLE_VERSIONS (king_phisher.constants.ConnectionErrorReason attribute), 72
ERROR_INVALID_CREDENTIALS (king_phisher.constants.ConnectionErrorReason attribute), 72
ERROR_INVALID_OTP (king_phisher.constants.ConnectionErrorReason attribute), 72
ERROR_PORT_FORWARD (king_phisher.constants.ConnectionErrorReason attribute), 72
ERROR_UNKNOWN (king_phisher.constants.ConnectionErrorReason attribute), 72
errors (module), 73
Event (class in king_phisher.server.web_sockets), 67
event_id (king_phisher.server.web_sockets.Event attribute), 67

event_socket (king_phisher.server.aaa.AuthenticatedSession attribute), 51
event_type (king_phisher.server.web_sockets.Event attribute), 67
event_type_filter() (in module king_phisher.client.server_events), 44
EventSocket (class in king_phisher.server.web_sockets), 67
ExceptionDialog (class in king_phisher.client.dialogs.exception), 6
exit() (built-in function), 102
expand_macros() (king_phisher.spf.SenderPolicyFramework method), 89
export_campaign_visit_geojson() (king_phisher.client.windows.main.MainAppWindow method), 19
export_campaign_xlsx() (king_phisher.client.windows.main.MainAppWindow method), 19
export_campaign_xml() (king_phisher.client.windows.main.MainAppWindow method), 19
export_database() (in module king_phisher.server.database.manager), 47
export_graph_provider() (in module king_phisher.client.graphs), 26
export_message_data() (king_phisher.client.tabs.mail.MailSenderTab method), 13
export_table_to_csv() (king_phisher.client.tabs.campaign.CampaignViewGenericTableTab method), 9
export_table_to_xlsx_worksheet() (king_phisher.client.tabs.campaign.CampaignViewGenericTableTab method), 9

F

file_name (client.web_cloner.ClonedResourceDetails attribute), 46
file_names (king_phisher.archive.ArchiveFile attribute), 70
FileChooserDialog (class in king_phisher.client.widget.extras), 13
FileMonitor (class in king_phisher.client.gui_utilities), 34
files (king_phisher.archive.ArchiveFile attribute), 70
files (king_phisher.client.mailer.MessageAttachments attribute), 40
finalize() (king_phisher.plugins.PluginBase method), 83
find (module), 74
first_name (king_phisher.client.mailer.MessageTarget attribute), 40
font_desc_italic (in module king_phisher.client.widget.resources), 17
ForkedAuthenticator (class in king_phisher.server.aaa), 52
format() (king_phisher.color.ColoredLogFormatter method), 72
format_cell_data() (king_phisher.client.tabs.campaign.CampaignViewGene method), 9
format_datetime() (in module king_phisher.utilities), 95
format_exception_details() (in module king_phisher.client.dialogs.exception), 5
format_exception_name() (in module king_phisher.client.dialogs.exception), 5
format_node_data() (king_phisher.client.tabs.campaign.CampaignViewGen method), 9
formatException() (king_phisher.color.ColoredLogFormatter static method), 72
formatted_description (king_phisher.plugins.PluginBaseMeta attribute), 83
FreezableDict (class in king_phisher.utilities), 97
freeze() (king_phisher.utilities.FreezableDict method), 97
from_db_authenticated_session() (king_phisher.server.aaa.AuthenticatedSession class method), 51
from_db_table_row_element() (in module king_phisher.serializers), 85
frozen (king_phisher.utilities.FreezableDict attribute), 97

G

generate_token() (in module king_phisher.server.rest_api), 56
geoip (module), 75
geoip_lookup() (king_phisher.client.client_rpc.KingPhisherRPCClient method), 23
geoip_lookup_multi() (king_phisher.client.client_rpc.KingPhisherRPCClien method), 23
GeoLocation (class in king_phisher.geoip), 76
get() (king_phisher.server.aaa.AuthenticatedSessionManager method), 51
get_active() (king_phisher.client.widget.managers.RadioButtonGroupMana method), 15
get_bind_addresses() (in module king_phisher.server.build), 53
get_client_ip() (king_phisher.server.server.KingPhisherRequestHandler method), 57
get_color() (king_phisher.client.graphs.GraphBase method), 27
get_data() (king_phisher.archive.ArchiveFile method), 70
get_entry_value() (king_phisher.client.gui_utilities.GladeGObject method), 35
get_file() (king_phisher.archive.ArchiveFile method), 70
get_graph() (in module king_phisher.client.graphs), 27
get_graphs() (in module king_phisher.client.graphs), 27
get_groups_for_user() (in module king_phisher.server.aaa), 51
get_meta_data() (in module king_phisher.server.database.manager), 48
get_mime_attachments() (king_phisher.client.mailer.MailSenderThread method), 38

get_popup_copy_submenu() (king_phisher.client.widget.managers.TreeViewManager method), 16
get_popup_menu() (king_phisher.client.widget.managers.TreeViewManager method), 16
get_proposal_terms() (in module king_phisher.client.widget.completion_providers) 17
get_query_creds() (king_phisher.server.server.KingPhisherRequestHandler method), 57
get_revision() (in module king_phisher.version), 98
get_row_by_id() (in module king_phisher.server.database.manager), 48
get_scale() (in module king_phisher.color), 71
get_smtp_servers() (in module king_phisher.sms), 87
get_ssl_hostnames() (in module king_phisher.server.build), 54
get_tables_with_column_id() (in module king_phisher.server.database.models), 49
get_tag_model() (king_phisher.client.client_rpc.KingPhisherRPCClient method), 23
get_template_vars_client() (king_phisher.server.server.KingPhisherRequestHandler method), 57
get_timedelta_for_offset() (in module king_phisher.ics), 76
get_tz_posix_env_var() (in module king_phisher.ics), 77
get_widget() (king_phisher.client.plugins.ClientOptionMixin method), 41
get_widget_value() (king_phisher.client.plugins.ClientOptionMixin method), 42
GladeDependencies (class in king_phisher.client.gui_utilities), 34
GladeGObject (class in king_phisher.client.gui_utilities), 35
GladeGObjectMeta (class in king_phisher.client.gui_utilities), 34
GladeGObjectMeta.assigned_name (class in king_phisher.client.gui_utilities), 35
GladeProxy (class in king_phisher.client.gui_utilities), 36
GladeProxyDestination (class in king_phisher.client.gui_utilities), 36
glib_idle_add_wait() (in module king_phisher.client.gui_utilities), 30
gobject_get_value() (in module king_phisher.client.gui_utilities), 30
GOBJECT_PROPERTY_MAP (in module king_phisher.client.gui_utilities), 30
gobject_signal_accumulator() (in module king_phisher.client.gui_utilities), 30
gobject_signal_blocked() (in module king_phisher.client.gui_utilities), 31
gobjects (king_phisher.client.gui_utilities.GladeGObject attribute), 35
graph_title (king_phisher.client.graphs.CampaignGraphMessageResults attribute), 28
graph_title (king_phisher.client.graphs.CampaignGraphOverview attribute), 28
graph_title (king_phisher.client.graphs.CampaignGraphPasswordComplexity attribute), 29
graph_title (king_phisher.client.graphs.CampaignGraphVisitorInfo attribute), 29
graph_title (king_phisher.client.graphs.CampaignGraphVisitorInfoPie attribute), 29
graph_title (king_phisher.client.graphs.CampaignGraphVisitsMap attribute), 29
graph_title (king_phisher.client.graphs.CampaignGraphVisitsTimeline attribute), 29
graph_title (king_phisher.client.graphs.GraphBase attribute), 27
GraphBase (class in king_phisher.client.graphs), 27
graphql() (king_phisher.client.client_rpc.KingPhisherRPCClient method), 23
graphs (king_phisher.client.tabs.campaign.CampaignViewDashboardTab attribute), 8
GTK3_DEFAULT_THEME (in module king_phisher.client.application), 20
gtk_builder (king_phisher.client.gui_utilities.GladeGObject attribute), 35
gtk_builder_get() (king_phisher.client.gui_utilities.GladeGObject method), 35
gtk_calendar_get_pydate() (in module king_phisher.client.gui_utilities), 31
gtk_calendar_set_pydate() (in module king_phisher.client.gui_utilities), 31
gtk_list_store_search() (in module king_phisher.client.gui_utilities), 31
gtk_menu_get_item_by_label() (in module king_phisher.client.gui_utilities), 31
gtk_menu_insert_by_path() (in module king_phisher.client.gui_utilities), 31
gtk_menu_position() (in module king_phisher.client.gui_utilities), 32
gtk_style_context_get_color() (in module king_phisher.client.gui_utilities), 32
gtk_sync() (in module king_phisher.client.gui_utilities), 32
gtk_treesortable_sort_func_numeric() (in module king_phisher.client.gui_utilities), 32
gtk_treeview_selection_iterate() (in module king_phisher.client.gui_utilities), 33
gtk_treeview_selection_to_clipboard() (in module king_phisher.client.gui_utilities), 32
gtk_treeview_set_column_titles() (in module king_phisher.client.gui_utilities), 33
gtk_widget_destroy_children() (in module king_phisher.client.gui_utilities), 33
guess_smtp_server_address() (in module

king_phisher.client.mailer), 37

H

has_file() (king_phisher.archive.ArchiveFile method), 70
has_matplotlib (in module king_phisher.client.graphs), 26
has_matplotlib_basemap (in module king_phisher.client.graphs), 26
has_vte (in module king_phisher.client.windows.rpc_terminal), 20
hash_algorithm (king_phisher.server.aaa.CachedPassword attribute), 52
headers (king_phisher.server.server.KingPhisherServer attribute), 58
homepage (king_phisher.plugins.PluginBase attribute), 83
HostKeyAcceptDialog (class in king_phisher.client.dialogs.ssh_host_key), 7
HostKeyWarnDialog (class in king_phisher.client.dialogs.ssh_host_key), 7
hosts() (king_phisher.ipaddress.IPv6Network method), 81
HTMLComletionProvider (in module king_phisher.client.widget.completion_providers), 17
http_request() (king_phisher.testing.KingPhisherServerTestCase method), 93

I

ics (module), 76
images (king_phisher.client.mailer.MessageAttachments attribute), 40
import_database() (in module king_phisher.server.database.manager), 47
import_message_data() (king_phisher.client.tabs.mail.MailSenderTab method), 13
ImportCampaignWindow (class in king_phisher.client.windows.campaign_import), 18
init_alembic() (in module king_phisher.server.database.manager), 48
init_data_path() (in module king_phisher.find), 75
init_database() (in module king_phisher.geoip), 75
init_database() (in module king_phisher.server.database.manager), 48
init_database_postgresql() (in module king_phisher.server.database.manager), 49
initialize() (king_phisher.plugins.PluginBase method), 83
IOS (king_phisher.constants.OSFamily attribute), 73
ip_address (king_phisher.geoip.GeoLocation attribute), 76
ip_address() (in module king_phisher.ipaddress), 78
ip_interface() (in module king_phisher.ipaddress), 79
ip_network() (in module king_phisher.ipaddress), 78
ipaddress (module), 78
ipv4_mapped (king_phisher.ipaddress.IPv6Address attribute), 80
IPv4Address (class in king_phisher.ipaddress), 79
IPv4Network (class in king_phisher.ipaddress), 80
IPv6Address (class in king_phisher.ipaddress), 80
IPv6Network (class in king_phisher.ipaddress), 81
is_archive() (in module king_phisher.archive), 69
is_available (king_phisher.client.graphs.CampaignGraphVisitsMap attribute), 29
is_compatible (king_phisher.plugins.PluginBaseMeta attribute), 83
is_connected (king_phisher.client.server_events.ServerEventSubscriber attribute), 45
is_global (king_phisher.ipaddress.IPv4Network attribute), 80
is_global (king_phisher.ipaddress.IPv6Address attribute), 80
is_link_local (king_phisher.ipaddress.IPv4Address attribute), 79
is_link_local (king_phisher.ipaddress.IPv6Address attribute), 80
is_loopback (king_phisher.ipaddress.IPv4Address attribute), 79
is_loopback (king_phisher.ipaddress.IPv6Address attribute), 80
is_loopback() (in module king_phisher.ipaddress), 79
is_multicast (king_phisher.ipaddress.IPv4Address attribute), 79
is_multicast (king_phisher.ipaddress.IPv6Address attribute), 80
is_private (king_phisher.ipaddress.IPv4Address attribute), 79
is_private (king_phisher.ipaddress.IPv6Address attribute), 80
is_private (king_phisher.server.database.models.BaseRowCls attribute), 50
is_reserved (king_phisher.ipaddress.IPv4Address attribute), 79
is_reserved (king_phisher.ipaddress.IPv6Address attribute), 80
is_site_local (king_phisher.ipaddress.IPv6Address attribute), 80
is_site_local (king_phisher.ipaddress.IPv6Network attribute), 81
is_subscribed() (king_phisher.client.server_events.ServerEventSubscriber method), 45
is_subscribed() (king_phisher.server.web_sockets.EventSocket method), 67
is_unspecified (king_phisher.ipaddress.IPv4Address attribute), 80
is_unspecified (king_phisher.ipaddress.IPv6Address attribute), 81

is_valid() (in module king_phisher.ipaddress), 79
is_valid_email_address() (in module king_phisher.utilities), 95
issue_alert() (king_phisher.server.server.KingPhisherRequestHandler method), 57
items() (king_phisher.constants.ConstantGroup class method), 72
iterations (king_phisher.server.aaa.CachedPassword attribute), 52

J

JinjaComletionProvider (in module king_phisher.client.widget.completion_providers), 18
JinjaEmailCompletionProvider (in module king_phisher.client.widget.completion_providers), 18
job_manager (king_phisher.server.server.KingPhisherServer attribute), 58
join_path() (king_phisher.templates.TemplateEnvironmentBase method), 92
JSON (class in king_phisher.serializers), 86

K

KeyValueStorage (class in king_phisher.server.database.storage), 50
KingPhisherAbortError, 73
KingPhisherAbortRequestError, 73
KingPhisherClientApplication (class in king_phisher.client.application), 21
KingPhisherDatabaseError, 73
KingPhisherError, 73
KingPhisherInputValidationError, 73
KingPhisherPermissionError, 73
KingPhisherPluginError, 73
KingPhisherRequestHandler (class in king_phisher.server.server), 57
KingPhisherResourceError, 74
KingPhisherRPCClient (class in king_phisher.client.client_rpc), 23
KingPhisherServer (class in king_phisher.server.server), 58
KingPhisherServerTestCase (class in king_phisher.testing), 93
KingPhisherSSHKeyError (class in king_phisher.ssh_forward), 91
KingPhisherTestCase (class in king_phisher.testing), 93
KingPhisherTimeoutError, 74
kwargs (king_phisher.client.gui_utilities.GladeProxyDestination attribute), 36

L

label (king_phisher.client.tabs.campaign.CampaignViewGenericTab attribute), 8
label (king_phisher.client.tabs.campaign.CampaignViewTab attribute), 10
label (king_phisher.client.tabs.mail.MailSenderConfigurationTab attribute), 11
label (king_phisher.client.tabs.mail.MailSenderEditTab attribute), 11
label (king_phisher.client.tabs.mail.MailSenderPreviewTab attribute), 11
label (king_phisher.client.tabs.mail.MailSenderSendTab attribute), 12
label (king_phisher.client.tabs.mail.MailSenderTab attribute), 13
label_text (king_phisher.client.tabs.campaign.CampaignViewDashboardTab attribute), 8
label_text (king_phisher.client.tabs.campaign.CampaignViewGenericTab attribute), 8
last_load_time (king_phisher.client.tabs.campaign.CampaignViewGenericT attribute), 8
last_name (king_phisher.client.mailer.MessageTarget attribute), 40
last_seen (king_phisher.server.aaa.AuthenticatedSession attribute), 51
latitude (king_phisher.geoip.Coordinates attribute), 75
line (king_phisher.client.mailer.MessageTarget attribute), 40
LINUX (king_phisher.constants.OSFamily attribute), 73
liststore_export() (in module king_phisher.client.export), 25
liststore_to_csv() (in module king_phisher.client.export), 26
liststore_to_xlsx_worksheet() (in module king_phisher.client.export), 26
load() (king_phisher.plugins.PluginManagerBase method), 84
load() (king_phisher.serializers.Serializer class method), 87
load_all() (king_phisher.plugins.PluginManagerBase method), 84
load_campaign_information() (king_phisher.client.tabs.campaign.CampaignViewDashboardTab method), 8
load_campaign_information() (king_phisher.client.tabs.campaign.CampaignViewGenericTableT method), 9
load_campaigns() (king_phisher.client.dialogs.campaign_selection.Campaig method), 4
load_campaigns() (king_phisher.client.windows.compare_campaigns.Camp method), 19
load_graph() (king_phisher.client.graphs.CampaignCompGraph method), 30
load_graph() (king_phisher.client.graphs.CampaignGraph method), 28
load_html_data() (king_phisher.client.widget.extras.WebKitHTMLView method), 14

load_html_file() (king_phisher.client.tabs.mail.MailSenderEditTab method), 11
load_html_file() (king_phisher.client.tabs.mail.MailSenderPreviewTab method), 12
load_html_file() (king_phisher.client.widget.extras.WebKitHTMLView method), 14
load_plugins() (king_phisher.client.windows.plugin_manager.PluginManagerWindow method), 20
load_server_config() (king_phisher.client.application.KingPhisherClientApplication method), 21
loaded_plugins (king_phisher.plugins.PluginManagerBase attribute), 84
loader_idle_routine() (king_phisher.client.tabs.campaign.CampaignViewGenericTableTab method), 8
loader_thread (king_phisher.client.tabs.campaign.CampaignViewGenericTableTab attribute), 8
loader_thread_lock (king_phisher.client.tabs.campaign.CampaignViewGenericTableTab attribute), 9
loader_thread_routine() (king_phisher.client.tabs.campaign.CampaignViewDashboardTab method), 8
loader_thread_routine() (king_phisher.client.tabs.campaign.CampaignViewGenericTableTab method), 9
loader_thread_stop (king_phisher.client.tabs.campaign.CampaignViewGenericTableTab attribute), 9
loads() (king_phisher.serializers.JSON class method), 86
loads() (king_phisher.serializers.MsgPack class method), 86
local_server (king_phisher.ssh_forward.SSHTCPForwarder attribute), 91
logger (king_phisher.server.web_sockets.WebSocketsManager attribute), 68
login() (king_phisher.client.client_rpc.KingPhisherRPCClient method), 23
LoginDialog (class in king_phisher.client.dialogs.login), 6
LoginDialogBase (class in king_phisher.client.dialogs.login), 6
longitude (king_phisher.geoip.Coordinates attribute), 75
lookup() (in module king_phisher.geoip), 75
lookup_carrier_gateway() (in module king_phisher.sms), 87

M

MACRO_REGEX (in module king_phisher.spf), 88
MailSenderConfigurationTab (class in king_phisher.client.tabs.mail), 11
MailSenderEditTab (class in king_phisher.client.tabs.mail), 11
MailSenderPreviewTab (class in king_phisher.client.tabs.mail), 11
MailSenderSendTab (class in king_phisher.client.tabs.mail), 12
MailSenderTab (class in king_phisher.client.tabs.mail), 12
MailSenderThread (class in king_phisher.client.mailer), 37
main_tab_window (king_phisher.client.application.KingPhisherClientApplication attribute), 21
MainAppWindow (class in king_phisher.client.windows.main), 19
MainMenuBar (class in king_phisher.client.windows.main), 19
make_child_page() (in module king_phisher.server.pages), 55
make_message_uid() (in module king_phisher.utilities), 95
make_redirect_page() (in module king_phisher.server.pages), 55
make_unique_id() (in module king_phisher.utilities), 95
make_window() (king_phisher.client.graphs.GraphBase method), 27
match (king_phisher.spf.SenderPolicyFramework attribute), 90
matches (king_phisher.spf.SenderPolicyFramework attribute), 90
MAX_QUERIES (in module king_phisher.spf), 88
MAX_VOID (in module king_phisher.spf), 88
merge_config() (king_phisher.client.application.KingPhisherClientApplication method), 21
message_data_to_kpm() (in module king_phisher.client.export), 25
message_id (king_phisher.server.server.KingPhisherRequestHandler attribute), 58
MessageAttachments (class in king_phisher.client.mailer), 39
MessageTarget (class in king_phisher.client.mailer), 40
MessageTemplateEnvironment (class in king_phisher.templates), 92
metadata_file_name (king_phisher.archive.ArchiveFile attribute), 70
method (king_phisher.client.gui_utilities.GladeProxyDestination attribute), 36
mime_type (client.web_cloner.ClonedResourceDetails attribute), 46
minimum_size (king_phisher.client.graphs.GraphBase attribute), 27
missing_files() (king_phisher.client.mailer.MailSenderThread method), 38
MissingHostKeyPolicy (class in king_phisher.client.dialogs.ssh_host_key), 7
Mock (class in king_phisher.utilities), 97
mode (king_phisher.archive.ArchiveFile attribute), 70
MODE_ANALYZE (king_phisher.templates.MessageTemplateEnvironment attribute), 92
MODE_PREVIEW (king_phisher.templates.MessageTemplateEnvironment attribute), 92

MODE_SEND (king_phisher.templates.MessageTemplateEnvironment attribute), 92
MsgPack (class in king_phisher.serializers), 86

N

name (king_phisher.client.graphs.CampaignGraphMessageResults attribute), 28
name (king_phisher.client.graphs.CampaignGraphOverview attribute), 28
name (king_phisher.client.graphs.CampaignGraphPasswordComplexityPie attribute), 29
name (king_phisher.client.graphs.CampaignGraphVisitorInfo attribute), 29
name (king_phisher.client.graphs.CampaignGraphVisitorInfoPie attribute), 29
name (king_phisher.client.graphs.CampaignGraphVisitsTimeline attribute), 29
name (king_phisher.client.graphs.GraphBase attribute), 27
name (king_phisher.client.gui_utilities.GladeDependencies attribute), 34
name (king_phisher.client.gui_utilities.GladeProxy attribute), 36
name (king_phisher.client.widget.resources.CompanyEditorGrid attribute), 17
name_human (king_phisher.client.graphs.CampaignGraphMessageResults attribute), 28
name_human (king_phisher.client.graphs.CampaignGraphOverview attribute), 28
name_human (king_phisher.client.graphs.CampaignGraphPasswordComplexityPie attribute), 29
name_human (king_phisher.client.graphs.CampaignGraphVisitorInfo attribute), 29
name_human (king_phisher.client.graphs.CampaignGraphVisitorInfoPie attribute), 29
name_human (king_phisher.client.graphs.CampaignGraphVisitsTimeline attribute), 30
name_human (king_phisher.client.graphs.GraphBase attribute), 28
names() (king_phisher.constants.ConstantGroup class method), 72
new_from_password() (king_phisher.server.aaa.CachedPassword class method), 52
node_query (king_phisher.client.tabs.campaign.CampaignViewGenericTableTab attribute), 10
normalize_connection_url() (in module king_phisher.server.database.manager), 48
notebook (king_phisher.client.tabs.campaign.CampaignViewTab attribute), 10
notebook (king_phisher.client.tabs.mail.MailSenderTab attribute), 13
notebook (king_phisher.client.windows.main.MainAppWindow attribute), 19
notify_sent() (king_phisher.client.tabs.mail.MailSenderSendTab method), 12
notify_status() (king_phisher.client.tabs.mail.MailSenderSendTab method), 12
notify_stopped() (king_phisher.client.tabs.mail.MailSenderSendTab method), 12

O

objects_load_from_config() (king_phisher.client.gui_utilities.GladeGObject method), 35
objects_persist (king_phisher.client.gui_utilities.GladeGObject attribute), 36
objects_save_to_config() (king_phisher.client.gui_utilities.GladeGObject method), 36
offset (king_phisher.ics.TimezoneOffsetDetails attribute), 78
offset_dst (king_phisher.ics.TimezoneOffsetDetails attribute), 78
open_uri() (in module king_phisher.utilities), 96
OptionBase (class in king_phisher.plugins), 81
OptionBoolean (class in king_phisher.plugins), 82
OptionEnum (class in king_phisher.plugins), 82
OptionInteger (class in king_phisher.plugins), 82
options (king_phisher.plugins.PluginBase attribute), 83
OptionString (class in king_phisher.plugins), 82
os_version (ua_parser.UserAgent attribute), 94
os_name (ua_parser.UserAgent attribute), 94
os_version_string (ua_parser.UserAgent attribute), 94
OSArch (class in king_phisher.constants), 72
OSFamily (class in king_phisher.constants), 73
OSX (king_phisher.constants.OSFamily attribute), 73

P

packed (king_phisher.ipaddress.IPv4Address attribute), 80
packed (king_phisher.ipaddress.IPv6Address attribute), 81
parent (king_phisher.client.gui_utilities.GladeGObject attribute), 36
parse_datetime() (in module king_phisher.utilities), 96
parse_tz_posix_env_var() (in module king_phisher.ics), 77
parse_user_agent() (in module king_phisher.ua_parser), 94
password_is_complex() (in module king_phisher.utilities), 96
patch_html() (king_phisher.client.web_cloner.WebPageCloner method), 47
path (king_phisher.server.server.KingPhisherRequestHandler attribute), 58
pause() (king_phisher.client.mailer.MailSenderThread method), 38

paused (king_phisher.client.mailer.MailSenderThread attribute), 38
ping() (king_phisher.client.client_rpc.KingPhisherRPCClient method), 23
ping_all() (king_phisher.server.web_sockets.WebSocketsManager method), 68
PluginBase (class in king_phisher.plugins), 82
PluginBaseMeta (class in king_phisher.plugins), 83
PluginManagerBase (class in king_phisher.plugins), 84
PluginManagerWindow (class in king_phisher.client.windows.plugin_manager), 20
plugins (module), 81
policy (king_phisher.spf.SenderPolicyFramework attribute), 90
popup_menu (king_phisher.client.tabs.campaign.CampaignViewGenericTableTab attribute), 10
postal_code (king_phisher.geoip.GeoLocation attribute), 76
PPC (king_phisher.constants.OSArch attribute), 72
preprep_xml_data() (king_phisher.client.windows.campaign_import.ImportCampaignWindow method), 18
print_error() (in module king_phisher.color), 71
print_good() (in module king_phisher.color), 71
print_status() (in module king_phisher.color), 71
process_attachment_file() (king_phisher.client.plugins.ClientPluginMailerAttachment method), 44
process_pause() (king_phisher.client.mailer.MailSenderThread method), 38
progressbar (king_phisher.client.tabs.mail.MailSenderSendTab attribute), 12
publish() (king_phisher.server.web_sockets.EventSocket method), 68
put() (king_phisher.server.aaa.AuthenticatedSessionManager method), 52
pw_hash (king_phisher.server.aaa.CachedPassword attribute), 52

Q

QUALIFIERS (in module king_phisher.spf), 89
Query (class in king_phisher.server.graphql), 54
quick_add_filter() (king_phisher.client.widget.extras.FileChooserDialog method), 14
quit() (king_phisher.client.application.KingPhisherClientApplication method), 21

R

RadioButtonGroupManager (class in king_phisher.client.widget.managers), 15
random_string() (in module king_phisher.utilities), 96
random_string_lower_numeric() (in module king_phisher.utilities), 96
reconnect (king_phisher.client.server_events.ServerEventSubscriber attribute), 45
reconnect() (king_phisher.client.client_rpc.KingPhisherRPCClient method), 24
records (king_phisher.spf.SenderPolicyFramework attribute), 90
references (king_phisher.client.application.KingPhisherClientApplication attribute), 22
refresh() (king_phisher.client.graphs.CampaignGraph method), 28
refresh_frequency (king_phisher.client.tabs.campaign.CampaignViewGener attribute), 9
register_http() (king_phisher.server.plugins.ServerPlugin method), 55
register_rpc() (in module king_phisher.server.server_rpc), 58
register_rpc() (king_phisher.server.plugins.ServerPlugin method), 56
register_table() (in module king_phisher.server.database.models), 49
reinitialize_rpc_clients() (king_phisher.client.client_rpc.KingPhisherRPCClien method), 24
remote_table() (king_phisher.client.client_rpc.KingPhisherRPCClient method), 24
remote_table_row() (king_phisher.client.client_rpc.KingPhisherRPCClient method), 24
RemoteRow (class in king_phisher.client.client_rpc), 24
remove() (king_phisher.server.aaa.AuthenticatedSessionManager method), 52
remove() (king_phisher.server.web_sockets.WebSocketsManager method), 68
remove_import_campaign() (king_phisher.client.windows.campaign_import.ImportCampaign method), 18
remove_office_metadata() (in module king_phisher.scrubber), 85
render_message_template() (in module king_phisher.client.mailer), 37
renderer_text_desc (in module king_phisher.client.widget.resources), 17
req_min_version (king_phisher.plugins.PluginBase attribute), 83
req_packages (king_phisher.plugins.PluginBase attribute), 83
request_received (in module king_phisher.server.signals), 66
resize() (king_phisher.client.graphs.GraphBase method), 28
resource (client.web_cloner.ClonedResourceDetails attribute), 46
resource_is_on_target() (king_phisher.client.web_cloner.WebPageCloner method), 47
response_sent (in module king_phisher.server.signals), 66
response_timeout (king_phisher.server.aaa.ForkedAuthenticator

attribute), 53
REST_API_BASE (in module king_phisher.server.rest_api), 56
rest_handler() (in module king_phisher.server.rest_api), 57
revision (in module king_phisher.version), 97
RFC
RFC 5545, 76
RFC 7208, 88–90
root_config (king_phisher.server.plugins.ServerPlugin attribute), 56
rpc (king_phisher.client.application.KingPhisherClientApplication attribute), 22
rpc (king_phisher.client.windows.main.MainAppWindow attribute), 19
rpc_api_version (in module king_phisher.version), 97
RPC_AUTH_HEADER (in module king_phisher.server.server_rpc), 58
rpc_campaign_alerts_is_subscribed() (in module king_phisher.server.server_rpc), 59
rpc_campaign_alerts_subscribe() (in module king_phisher.server.server_rpc), 59
rpc_campaign_alerts_unsubscribe() (in module king_phisher.server.server_rpc), 59
rpc_campaign_landing_page_new() (in module king_phisher.server.server_rpc), 59
rpc_campaign_message_new() (in module king_phisher.server.server_rpc), 59
rpc_campaign_new() (in module king_phisher.server.server_rpc), 60
rpc_campaign_stats() (in module king_phisher.server.server_rpc), 60
rpc_config_get() (in module king_phisher.server.server_rpc), 60
rpc_config_set() (in module king_phisher.server.server_rpc), 60
rpc_database_count_rows() (in module king_phisher.server.server_rpc), 61
rpc_database_delete_row_by_id() (in module king_phisher.server.server_rpc), 61
rpc_database_delete_rows_by_id() (in module king_phisher.server.server_rpc), 61
rpc_database_get_row_by_id() (in module king_phisher.server.server_rpc), 61
rpc_database_insert_row() (in module king_phisher.server.server_rpc), 62
rpc_database_set_row_value() (in module king_phisher.server.server_rpc), 62
rpc_database_view_rows() (in module king_phisher.server.server_rpc), 62
rpc_events_is_subscribed() (in module king_phisher.server.server_rpc), 60
rpc_events_subscribe() (in module king_phisher.server.server_rpc), 60
rpc_events_unsubscribe() (in module king_phisher.server.server_rpc), 61
rpc_geoip_lookup() (in module king_phisher.server.server_rpc), 62
rpc_geoip_lookup_multi() (in module king_phisher.server.server_rpc), 63
rpc_graphql() (in module king_phisher.server.server_rpc), 63
rpc_login() (in module king_phisher.server.server_rpc), 63
rpc_logout() (in module king_phisher.server.server_rpc), 63
rpc_method_call (in module king_phisher.server.signals), 66
rpc_method_called (in module king_phisher.server.signals), 66
rpc_ping() (in module king_phisher.server.server_rpc), 63
rpc_plugins_list() (in module king_phisher.server.server_rpc), 63
rpc_shutdown() (in module king_phisher.server.server_rpc), 63
rpc_user_logged_in (in module king_phisher.server.signals), 66
rpc_user_logged_out (in module king_phisher.server.signals), 66
rpc_version() (in module king_phisher.server.server_rpc), 63
RPCTerminal (class in king_phisher.client.windows.rpc_terminal), 20
run_quick_open() (king_phisher.client.widget.extras.FileChooserDialog method), 14
run_quick_save() (king_phisher.client.widget.extras.FileChooserDialog method), 14
run_quick_select_directory() (king_phisher.client.widget.extras.FileChooserDialog method), 14
running (king_phisher.client.mailer.MailSenderThread attribute), 38

S

safe_send() (in module king_phisher.server.signals), 64
salt (king_phisher.server.aaa.CachedPassword attribute), 52
save_html_file() (king_phisher.client.tabs.mail.MailSenderEditTab method), 11
Schema (class in king_phisher.server.graphql), 54
SCHEMA_VERSION (in module king_phisher.server.database.models), 49
scrubber (module), 85
select_xml_campaign() (king_phisher.client.windows.campaign_import.Imp method), 18
send() (king_phisher.server.aaa.ForkedAuthenticator method), 53

send_message() (king_phisher.client.mailer.MailSenderThread method), 39
send_sms() (in module king_phisher.sms), 88
sender_start_failure() (king_phisher.client.tabs.mail.MailSenderSendTab method), 12
sender_thread (king_phisher.client.tabs.mail.MailSenderSendTab attribute), 12
SenderPolicyFramework (class in king_phisher.spf), 89
sequence_number (king_phisher.server.aaa.ForkedAuthenticator attribute), 53
Serializer (class in king_phisher.serializers), 86
serializer (king_phisher.server.database.storage.KeyValueStorage attribute), 50
serializers (module), 85
serve_forever() (king_phisher.smtp_server.BaseSMTPServer method), 88
server (king_phisher.server.plugins.ServerPlugin attribute), 56
server.aaa (module), 51
server.build (module), 53
server.graphql (module), 54
server.pages (module), 55
server.plugins (module), 55
server.rest_api (module), 56
server.server (module), 57
server.server_rpc (module), 58
server.signals (module), 64
server.web_sockets (module), 67
server_events (king_phisher.client.application.KingPhisherClientApplication attribute), 22
server_from_config() (in module king_phisher.server.build), 54
server_initialized (in module king_phisher.server.signals), 67
server_smtp_connect() (king_phisher.client.mailer.MailSenderThread method), 39
server_smtp_disconnect() (king_phisher.client.mailer.MailSenderThread method), 39
server_smtp_reconnect() (king_phisher.client.mailer.MailSenderThread method), 39
server_ssh_connect() (king_phisher.client.mailer.MailSenderThread method), 39
ServerEventSubscriber (class in king_phisher.client.server_events), 45
ServerPlugin (class in king_phisher.server.plugins), 55
ServerPluginManager (class in king_phisher.server.plugins), 56
session_has_permissions() (king_phisher.server.database.models.BaseRowCls method), 50
set_active() (king_phisher.client.widget.managers.RadioButtonGroupManager method), 15
set_column_color() (king_phisher.client.widget.managers.TreeViewManager method), 16
set_column_titles() (king_phisher.client.widget.managers.TreeViewManager method), 16
set_schema_data() (in module king_phisher.server.database.manager), 49
set_mode() (king_phisher.templates.MessageTemplateEnvironment method), 92
show_campaign_graph() (king_phisher.client.application.KingPhisherClient method), 22
show_campaign_selection() (king_phisher.client.application.KingPhisherClientApplication method), 22
show_dialog() (in module king_phisher.client.gui_utilities), 33
show_dialog_error() (in module king_phisher.client.gui_utilities), 34
show_dialog_exc_socket_error() (in module king_phisher.client.gui_utilities), 34
show_dialog_info() (in module king_phisher.client.gui_utilities), 34
show_dialog_warning() (in module king_phisher.client.gui_utilities), 34
show_dialog_yes_no() (in module king_phisher.client.gui_utilities), 34
show_preferences() (king_phisher.client.application.KingPhisherClientApp method), 22
show_tab() (king_phisher.client.tabs.mail.MailSenderEditTab method), 11
show_tab() (king_phisher.client.tabs.mail.MailSenderPreviewTab method), 12
shutdown() (king_phisher.client.server_events.ServerEventSubscriber method), 45
shutdown() (king_phisher.plugins.PluginManagerBase method), 84
shutdown() (king_phisher.server.server.KingPhisherServer method), 58
signal_button_pressed() (king_phisher.client.widget.extras.WebKitHTMLVi method), 15
signal_connect() (king_phisher.client.plugins.ClientPlugin method), 43
signal_connect_server_event() (king_phisher.client.plugins.ClientPlugin method), 43
signal_decide_policy() (king_phisher.client.widget.extras.WebKitHTMLVie method), 15
signal_decide_policy_webkit() (king_phisher.client.widget.extras.WebKitHTMLView method), 15
signal_entry_change() (king_phisher.client.windows.campaign_import.Impo method), 18
signal_import_button() (king_phisher.client.windows.campaign_import.Imp method), 18
signal_window_delete_event() (king_phisher.client.windows.campaign_import.ImportCampaign

method), 18
sixtofour (king_phisher.ipaddress.IPv6Address attribute), 81
size (client.web_cloner.ClonedResourceDetails attribute), 46
sms (module), 87
smtp_connection (king_phisher.client.mailer.MailSenderThread attribute), 39
smtp_server (module), 88
SMTPLoginDialog (class in king_phisher.client.dialogs.login), 6
sources (king_phisher.server.web_sockets.Event attribute), 67
spf (module), 88
SPFDirective (class in king_phisher.spf), 90
SPFError, 90
SPFMatch (class in king_phisher.spf), 90
SPFParseError, 90
SPFPermError, 90
SPFRecord (class in king_phisher.spf), 90
SPFTempError, 90
ssh_forward (module), 91
SSHLoginDialog (class in king_phisher.client.dialogs.login), 6
SSHTCPForwarder (class in king_phisher.ssh_forward), 91
standard_variables (king_phisher.templates.TemplateEnvironmentBase attribute), 92
start_process() (in module king_phisher.utilities), 96
stop() (king_phisher.client.mailer.MailSenderThread method), 39
stop() (king_phisher.server.aaa.AuthenticatedSessionManager method), 52
stop() (king_phisher.server.aaa.ForkedAuthenticator method), 53
stop() (king_phisher.server.web_sockets.WebSocketsManager method), 69
stop_cloning() (king_phisher.client.web_cloner.WebPageCloner method), 47
stop_remote_service() (king_phisher.client.application.KingPhisherClientApplication method), 22
storage (king_phisher.server.plugins.ServerPlugin attribute), 56
subscribe() (king_phisher.client.server_events.ServerEventSubscriber method), 45
subscribe() (king_phisher.server.web_sockets.EventSocket method), 68
SUCCESS (king_phisher.constants.ConnectionErrorReason attribute), 72
switch() (in module king_phisher.utilities), 96

T

tab (king_phisher.client.mailer.MailSenderThread attribute), 39
tab_notify_sent() (king_phisher.client.mailer.MailSenderThread method), 39
tab_notify_status() (king_phisher.client.mailer.MailSenderThread method), 39
tab_notify_stopped() (king_phisher.client.mailer.MailSenderThread method), 39
table_name (king_phisher.client.tabs.campaign.CampaignViewGenericTable attribute), 10
table_query (king_phisher.client.tabs.campaign.CampaignViewGenericTable attribute), 10
table_subscriptions (king_phisher.client.graphs.CampaignGraphMessageResult attribute), 28
table_subscriptions (king_phisher.client.graphs.CampaignGraphOverview attribute), 28
table_subscriptions (king_phisher.client.graphs.CampaignGraphPasswordCount attribute), 29
table_subscriptions (king_phisher.client.graphs.CampaignGraphVisitorInfo attribute), 29
table_subscriptions (king_phisher.client.graphs.CampaignGraphVisitorInfoTimeline attribute), 29
table_subscriptions (king_phisher.client.graphs.CampaignGraphVisitsMap attribute), 29
table_subscriptions (king_phisher.client.graphs.CampaignGraphVisitsTimeline attribute), 30
table_subscriptions (king_phisher.client.graphs.GraphBase attribute), 28
tabs (king_phisher.client.tabs.campaign.CampaignViewTab attribute), 10
tabs (king_phisher.client.tabs.mail.MailSenderTab attribute), 13
tabs.campaign (module), 8
tabs.mail (module), 10
TagEditorDialog (class in king_phisher.client.dialogs.tag_editor), 7
target_file (king_phisher.client.mailer.MailSenderThread attribute), 39
TemplateEnvironmentBase (class in king_phisher.templates), 92
templates (module), 91
teredo (king_phisher.ipaddress.IPv6Address attribute), 81
TEST_MESSAGE_TEMPLATE (in module king_phisher.testing), 92
TEST_MESSAGE_TEMPLATE_INLINE_IMAGE (in module king_phisher.testing), 92
test_webserver_url() (in module king_phisher.client.tabs.mail), 11
testing (module), 92
text_insert() (king_phisher.client.tabs.mail.MailSenderSendTab method), 12
textbuffer (king_phisher.client.tabs.mail.MailSenderEditTab attribute), 11
textbuffer (king_phisher.client.tabs.mail.MailSenderSendTab attribute), 12
TextEntryDialog (class in

king_phisher.client.dialogs.entry), 5
textview (king_phisher.client.tabs.mail.MailSenderEditTab attribute), 11
textview (king_phisher.client.tabs.mail.MailSenderSendTab attribute), 12
thaw() (king_phisher.utilities.FreezableDict method), 97
time (king_phisher.server.aaa.CachedPassword attribute), 52
time_zone (king_phisher.geoip.GeoLocation attribute), 76
Timezone (class in king_phisher.ics), 78
TimezoneOffsetDetails (class in king_phisher.ics), 78
title (king_phisher.plugins.PluginBase attribute), 83
to_elementtree_subelement() (in module king_phisher.serializers), 85
top_gobject (king_phisher.client.gui_utilities.GladeGObject attribute), 36
top_level (king_phisher.client.gui_utilities.GladeDependencies attribute), 34
TopMIMEMultipart (class in king_phisher.client.mailer), 40
treeview (king_phisher.client.widget.managers.TreeViewManager attribute), 16
TreeViewManager (class in king_phisher.client.widget.managers), 15

U

ua_parser (module), 93
uid (king_phisher.client.mailer.MessageTarget attribute), 40
unload() (king_phisher.plugins.PluginManagerBase method), 85
unload_all() (king_phisher.plugins.PluginManagerBase method), 85
unpause() (king_phisher.client.mailer.MailSenderThread method), 39
UNRESOLVED (in module king_phisher.client.client_rpc), 22
unsubscribe() (king_phisher.client.server_events.ServerEventSubscriber method), 46
unsubscribe() (king_phisher.server.web_sockets.EventSocket method), 68
user (king_phisher.server.aaa.AuthenticatedSession attribute), 51
USER_DATA_PATH (in module king_phisher.client.application), 20
UserAgent (class in king_phisher.ua_parser), 94
utilities (module), 94

V

validate_record() (in module king_phisher.spf), 89
values() (king_phisher.constants.ConstantGroup class method), 72
version (in module king_phisher.version), 98
version (king_phisher.plugins.PluginBase attribute), 83
version (module), 97
version_info (in module king_phisher.version), 98
version_label (in module king_phisher.version), 98
vhost (king_phisher.server.server.KingPhisherRequestHandler attribute), 58
view_columns (king_phisher.client.tabs.campaign.CampaignViewGenericTa attribute), 10
VIEW_ROW_COUNT (in module king_phisher.server.server_rpc), 59
visit_id (king_phisher.server.server.KingPhisherRequestHandler attribute), 58
visit_received (in module king_phisher.server.signals), 67
vte_child_routine() (in module king_phisher.client.client_rpc), 22

W

wait() (king_phisher.client.web_cloner.WebPageCloner method), 47
web_root_files() (king_phisher.testing.KingPhisherServerTestCase method), 93
WebKitHTMLView (class in king_phisher.client.widget.extras), 14
WebPageCloner (class in king_phisher.client.web_cloner), 46
WebSocketsManager (class in king_phisher.server.web_sockets), 68
webview (king_phisher.client.tabs.mail.MailSenderPreviewTab attribute), 12
which_glade() (in module king_phisher.client.gui_utilities), 34
widget (king_phisher.client.gui_utilities.GladeProxyDestination attribute), 36
widget.completion_providers (module), 17
widget.extras (module), 13
widget.managers (module), 15
widget.resources (module), 17
WINDOWS (king_phisher.constants.OSFamily attribute), 73

X

X86 (king_phisher.constants.OSArch attribute), 72
X86_64 (king_phisher.constants.OSArch attribute), 72
xor (module), 98
xor_decode() (in module king_phisher.xor), 98
xor_encode() (in module king_phisher.xor), 98

Z

zoneinfo_path (in module king_phisher.ics), 76