Zabbix-AI Integration Platform Documentation

Introduction

The provided Python script serves as an essential interface between the Zabbix Monitoring System and an AI Agent (currently utilizes OpenAI's GPT-3.5 Turbo model). In a world where timely and effective monitoring is critical, **this tool aims to revolutionize the way we manage system issues**.

- **Objective: The primary goal of this automation is to streamline the process of identifying problems and suggesting possible solutions**. When Zabbix detects an issue, the script collects the details, reviews past solutions, consults the AI Agent for recommendations, and then acknowledges the issue in Zabbix, thereby informing the team of a potential solution.
- **Context:** This automation is particularly beneficial for UNDP, an organization with a global footprint, spanning multiple time zones and offices in various countries. **The need for 24/7 monitoring is obvious, and this tool is designed to handle the most basic, initial tasks, thereby reducing the team's workload in responding to country offices.**
- **Philosophy:** It is crucial to emphasize that the **AI's role is not to replace human intelligence but to assist them**. In this framework, AI assists by quickly identifying issues and providing immediate feedback to country offices. It gathers and summarizes historical data and rapidly suggests solutions based on its extensive database**. This not only eliminates the need for redundant explanations but also help the team from the burden of repetitive initial responses.**
- **Human Oversight:** While AI will handle the initial analysis and recommendation, the **final decision on issue resolution remains in the hands of the human team**. This ensures that complex, specific problems are addressed with the level of scrutiny and expertise that only humans can provide.

Requirements

Software and Libraries:

- **Python Version:** Must be Python 3.0 or higher. The code uses syntax and libraries that are compatible with Python 3.x.
- **Requests Library:** This Python library is used for making HTTP requests to both Zabbix and AI APIs. It can be installed using pip: *pip install requests*.

External Services

- **Zabbix Server:** There must have access to a Zabbix server. Ensure that the Zabbix server is running and reachable. This is needed to monitor ICT infrastructure.
- **API Token:** A Zabbix API token (ZABBIX_API_TOKEN) is required for authentication. This token should have permissions to read hosts, host groups, and events, as well as to acknowledge events.
- **AI API:** The code also requires access to AI API for generating textual solutions based on event data.

- **AI API Token:** An AI API token (GPT_API_TOKEN) is required for authentication. You can get this by signing up for OpenAI's API services.

Installation

Opening of Command Line Interface (CLI) or Terminal is Required:

- On Windows, either the Command Prompt or PowerShell can be used.
- On macOS and Linux, the Terminal is to be opened.

Activation of the Virtual Environment (Optional):

- If a Python virtual environment is used, it is to be activated before proceeding.

  source <path_to_virtual_environment>/bin/activate # On macOS/Linux

  .\<path_to_virtual_environment>\Scripts\Activate # On Windows

Execution of Installation Command is Required:

- The following command must be run to install the necessary Python package.

  pip install <required_package_name>

Configuration

After the required Python package has been installed, several variables are to be configured:

Zabbix Configuration

- ***ZABBIX_URL*:** The URL of the Zabbix server's API is to be found. This URL is usually located on the Zabbix server dashboard and follows the format http://<the_zabbix_server_address>/api_jsonrpc.php. This variable is to be set in the script or as an environment variable.
- ***ZABBIX_API_TOKEN*:** An API token for authentication against the Zabbix server is to be generated. This token can usually be created in the API settings of the Zabbix dashboard. Once generated, this token is to be set in the script or as an environment variable.

AI Agent Configuration

- ***GPT_API_URL*:** The URL of the AI agent's API is to be included. This URL is generally provided by OpenAI and is to be set in the script or as an environment variable.
- ***GPT_API_TOKEN*:** An API token for authentication against the AI agent's API is to be obtained. This token is provided upon registration for the AI agent's API service. Safe storage of this token is advised, and it is to be set in the script or as an environment variable.

How to Run

Run the Python script with the following command: python zabbix_ai.py

Functions:

get_hostids_by_group(token, group_id):

- **Purpose:** Fetches the host IDs that belong to a specified group in Zabbix.
- **Parameters:**
  - ***token:*** API token for authenticating with Zabbix.
  - ***group_id:*** ID of the host group in Zabbix.
- **Returns:** A list of host IDs belonging to the specified group.

print_group_name(token, group_id):

- **Purpose:** Retrieves and prints the name of the host group based on its ID.
- **Parameters:**
  - ***token:*** API token for authenticating with Zabbix.
  - ***group_id:*** ID of the host group in Zabbix.
- **Return:** The name of the host group to the console.

get_problems(token, group_id, host_ids):

- **Purpose:** Retrieves the current unresolved problem events from Zabbix associated with the specified host group.
- **Parameters:**
  - ***token*:** API token for authenticating with Zabbix.
  - ***group_id:*** ID of the host group in Zabbix.
  - ***host_ids:*** List of host IDs for filtering the problems.
- **Returns:** A tuple containing a list of problem events and a unique timestamp ID.

get_host_inventory(token, host_id):

- **Purpose:** Fetches and retrieves the inventory notes associated with a specific host in Zabbix.
- **Parameters:**
  - ***token:*** API token for authenticating with Zabbix.
  - ***host_id:*** ID of the host in Zabbix.
- **Returns:** Inventory notes of the specified host, or an empty string if not found.

confirm_hostid(token, event_id):

- **Purpose:** Validates and confirms the host ID associated with a given event ID.
- **Parameters:**
  - ***token:*** API token for authenticating with Zabbix.
  - ***event_id:*** ID of the event in Zabbix.
- **Returns:** A list containing information about the host associated with the event.

get_past_solutions(token, host_id, object_id, id_num):

- **Purpose:** Retrieves past acknowledged events/solutions for a particular host and trigger combination.
- **Parameters:**
  - ***token:*** API token for authenticating with Zabbix.
  - ***host_id:*** ID of the host in Zabbix.

- ***object_id:*** ID of the trigger/event object.
- ***id_num:*** A unique identifier used for the API request.

- **Returns:** A list of past events with acknowledged solutions/messages.

process_events(events_list):

- **Purpose:** Transforms raw event data from Zabbix into a more readable format.
- **Parameters:**
  - ***events_list:*** List of raw event data fetched from Zabbix.
- **Returns:** A list of processed events with key information like timestamp and acknowledgement messages.

get_gpt_response(warning_message):

- **Purpose**: Sends a warning message to the GPT-3.5 model and gets a potential solution/response.
- **Parameters:**
  - ***warning_message:*** Message detailing the Zabbix warning or issue.
- **Returns:** The generated response/solution from GPT-3.5.

acknowledge_event(token, event_id, message, id_num):

- **Purpose:** Sends an acknowledgement for an event in Zabbix with a provided message.
- **Parameters:**
  - ***token:*** API token for authenticating with Zabbix.
  - ***event_id:*** ID of the event to be acknowledged in Zabbix.
  - ***message:*** Acknowledgement message to attach to the event.
  - ***id_num:*** A unique identifier used for the API request**.**
- **Return:** Success or failure message indicating the result of the acknowledgement request.

Execution Logic

Initial Setup

The program commences by configuring essential elements such as Zabbix and AI Agent API tokens, endpoints, and HTTP headers to facilitate proper communication.

Monitoring

1. **Fetching Host IDs:** At the outset, the program queries Zabbix for a roster of host IDs that are part of a designated group.

2. **Fetching Current Events:** Following this, the program retrieves ongoing, unresolved problems associated with the hosts from Zabbix. Each event comes with specifics like event ID, object ID (trigger ID), severity, and acknowledgment status.

3. **Event Processing:** For every unacknowledged event, the program executes the following actions:

   - **Confirm Host Info:** The application retrieves supplementary details about the host tied to the event, such as host ID and any inventory notes.

- **Past Solutions Retrieval:** For the current host and associated problem, the program solicits Zabbix for previously acknowledged solutions. This data is then parsed and made more readable.
- **Prompt Generation for AI Agent:** Depending on the current event details and any past solutions, a unique prompt is formulated and dispatched to the AI Agent API. The content of the prompt may differ based on the availability of previous solutions.
- **AI Agent Consultation:** The formulated prompt is relayed to the AI Agent, which in return proposes a suggested course of action or remedy for the problem at hand.
- **Acknowledge Event:** Ultimately, the program recognizes the event in Zabbix and appends the AI Agent-proposed solution to it.

Conclusion

Thank you for going through this comprehensive guide on the Zabbix-AI Integration Script. This tool is an innovative solution designed to bridge the Zabbix Monitoring System with AI capabilities. Our aim is to automate the initial stages of issue identification and resolution, effectively making the task more efficient and less labor-intensive for your teams, especially those that operate on a global scale like the UNDP.

Key Takeaways:

- **Streamlined Monitoring**: This script takes over the monotonous, initial tasks of identifying issues and suggesting solutions, thereby freeing up your human resources to focus on more complex problems.
- **Augmented Decision-making**: While the AI agent provides immediate, data-driven suggestions, it's designed to assist rather than replace human decision-makers.
- **Global Reach**: Especially beneficial for organizations like the UNDP, which require round-the-clock monitoring across multiple locations worldwide.

Next Steps:

- **Download and Configuration**: Ensure all prerequisites are met and carefully follow the installation and configuration steps.
- **Run and Monitor**: Once set up, keep an eye on the logs and notifications to ensure the script is performing as expected.
- **Leveraging Open Source**: We are actively looking into integrating open-source solutions like Facebook's Llama to train our specialized AI, aiming to enhance the capability and security of the data involved.
- **Data Security**: We understand the critical importance of data security, especially when dealing with sensitive IT monitoring data. We will implement multi-layer encryption and access control measures to protect the integrity and confidentiality of the data.
- **Continuous Improvement**: Given the dynamic nature of network environments and AI models, it's advisable to regularly update the script and its configurations.

Your feedback and contributions to this tool are invaluable for its ongoing improvement and effectiveness. Thank you for your interest and good luck with implementing this Zabbix-AI Integration Script into your monitoring workflow.