Workday
ThousandEyes Script - Functional Specification

Change Control

Title:	ThousandEyes Configuration Script
Version:	1.0
Date of version:	December 1, 2025
Created by:	
Confidentiality level:	Confidential

Version History

Date	**Version**	**Created By**	**Description of Change**
December 1, 2025	0.1	Daniela Magallan	Initial Draft

Resources

The following parties are involved in the engagement.

Name	Contact	Company	Title
Daniela Magallan	dmagalla@cisco.com	ThousandEyes	Integration Engineer
Fernanda Ramirez	ferramir@cisco.com	ThousandEyes	Integration Engineer

Description

This document serves as the functional specification for the script delivered by ThousandEyes to the Workday account.

Licensing

All engagements which require ThousandEyes Professional Services or TAM to create code for the Client are licensed under the Client's MSA with ThousandEyes. If open-source software is required, ThousandEyes will notify Client of such inclusions and document it appropriately in the final report.

Table of Contents

Contents

Introduction

Purpose

The script interacts with the ThousandEyes API to retrieve existing test IDs and automatically generate the corresponding Terraform import blocks. This enables the seamless import and management of existing tests through Terraform.

This solution is designed to reduce manual intervention in managing test configurations, improving reliability and efficiency while allowing the organization to focus on more critical tasks.

ThousandEyes Solution

The following sections describe all deliverables involved for the solution.

Solution Components

Python Script(s)

The script involved in this solution is written in Python, using 3.12.11 as the target version. The script was developed using Visual Studio Code IDE on MacOS platform.

Terraform project

The script generates two output files, “imports.tf” and “providers.tf”. These Terraform files are later used to import existing resources and generate the corresponding Terraform configuration files based on the retrieved information.

Script Overview

The script consists of two main components, described below. You will see how they connect with the configuration in the following sections.

1. `main_endpoint.py`:
    - Serves as the primary entry point.
    - Utilizes the **colorama** library to guide the user through the console interface and required steps, such as selecting account groups.
    - Logs key events through a custom logging service to support debugging and auditing.
2. `.env File`:

- This file contains essential configuration variables that define the behavior of the script. It includes:

  - **API_TOKEN**: A bearer token generated from the platform for API authentication.
    Example: API_TOKEN='dd2f410d-24fa8-829e6593d06'
  - **ORG_NAME**: The name of the customer's organization.
    Example: ORG_NAME='TE: GA_PLUS Agent Fix Org'
  - **TERRAFORM_PROJECT_PATH**: Path to the folder where Terraform files will be generated and Terraform commands will be executed.
    Example: TERRAFORM_PROJECT_PATH="./tf_project"
  - **TE_TF_VERSION**: Specifies the version of the ThousandEyes Terraform provider to be downloaded when initializing the Terraform project.
    Example: TE_TF_VERSION=">= 3.1.0"

- The script reads these variables to establish API connections, define the execution scope, and ensure logging occurs under the correct organizational context.

The script is modular, allowing customization and extension. It is designed with scalability and maintainability in mind, making it suitable for organizations managing a wide range of automated tests.

Requirements

API Token

To retrieve account group configurations, you must use a user account with Organization Admin privileges. After identifying the appropriate user, you will need to obtain its API Bearer Token, which is required for authentication when accessing the ThousandEyes API.

If you are not sure how to generate or retrieve the API token, please refer to the official ThousandEyes documentation or your organization's internal guidelines.
https://developer.cisco.com/docs/thousandeyes/authentication/#obtaining-your-bearer-token

Install Python

1. Download Python:
   - Visit the official Python website: https://www.python.org/downloads/
   - Download the latest stable version compatible with the application (preferably Python 3.9 or higher).
2. Install Python:
   - Run the installer.
   - Check the box **"Add Python to PATH"** before proceeding.
   - Choose "Customize installation" and ensure **pip**, **venv**, and **tcl/tk** are selected.
   - Complete the installation.

3. Verify Python Installation:
   - Open Command Prompt and run:

     ```
     python --version
     pip --version
     ```

Install Terraform

1. Go to the official HashiCorp Terraform download page and locate the version appropriate for your operating system. https://developer.hashicorp.com/terraform/install
2. Download the Terraform package for your OS (Windows, macOS, or Linux).
3. Extract the downloaded archive. It contains a single executable named terraform.
4. Move the terraform executable to a directory that is included in your system's PATH environment variable. This allows you to run terraform from any terminal.
5. Verify the installation by opening a terminal and running: terraform version
6. If the version is displayed correctly, Terraform is successfully installed and ready to use.

Install Required Libraries

1. (Optional) Create a virtual environment to isolate your project dependencies from the system installation. This helps prevent version conflicts and keeps the environment clean.
2. (Optional) Activate the virtual environment according to your operating system's instructions.
3. Ensure you are in the project directory where the python project and the **requirements.txt** file is located.
4. Install all required Python dependencies by running the following command:

   ```
   pip install -r requirements.txt
   ```

5. Once the installation is complete, your environment will have all the libraries needed for the script to run properly.

Mode of Operation

Step 1: Create a .env file and add your information.

1. Create a file named .env in the project directory.
2. Add the following variables to the file with your own information, do not copy and paste, this is just an example:

```
API_TOKEN='2XX6cc22-9dXXX-2d-9469b4d33c63d'
ORG_NAME="Dani's sandbox"
TERRAFORM_PROJECT_PATH= "./tf_project"
TE_TF_VERSION=">= 3.1.0"
```

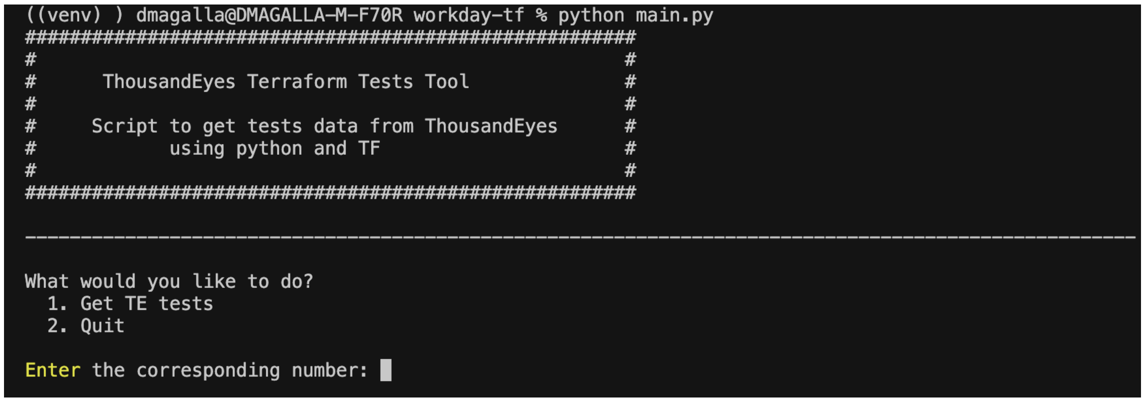
Note: The `TERRAFORM_PROJECT_PATH` specifies the local directory where the Terraform files will be generated. Make sure this path points to the folder where you want the project to be created and managed.

Step 2: Execute python script.

1. (Optional) If you created a virtual environment and it is not currently active, activate it first by following the instructions for your operating system.

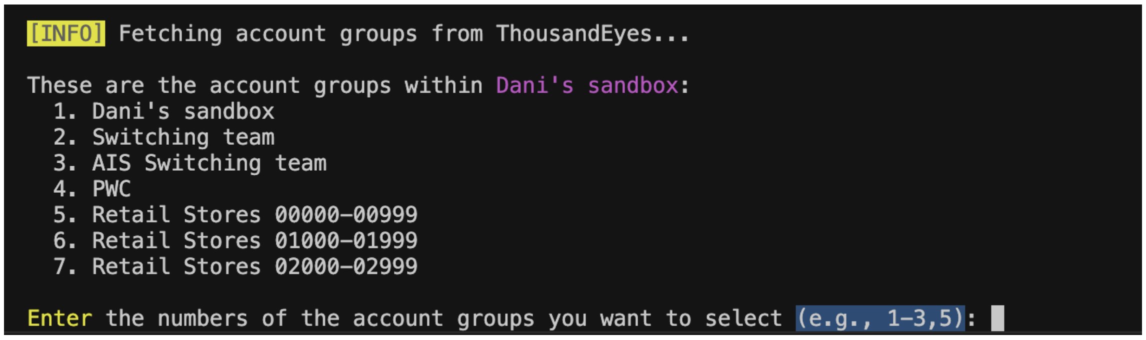
2. If you are not already in the project directory, navigate to it. Once inside that folder, run the following command to start the script:
   ```
   python main.py
   ```

3. If you are ready to retrieve test information from your ThousandEyes account, select option 1.



```
((venv) ) dmagalla@DMAGALLA-M-F70R workday-tf % python main.py
#######################################################
#                                                     #
#      ThousandEyes Terraform Tests Tool              #
#                                                     #
#     Script to get tests data from ThousandEyes      #
#            using python and TF                      #
#                                                     #
#######################################################

----------------------------------------------------------------------------------------------------

What would you like to do?
  1. Get TE tests
  2. Quit

Enter the corresponding number: █
```

4. Next, choose the account groups from which you want to fetch the data. You may select a continuous range using a hyphen or specify individual groups separated by commas (e.g., 1-3,5). After entering your selection, confirm it to proceed.



```
[INFO] Fetching account groups from ThousandEyes...

These are the account groups within Dani's sandbox:
  1. Dani's sandbox
  2. Switching team
  3. AIS Switching team
  4. PWC
  5. Retail Stores 00000-00999
  6. Retail Stores 01000-01999
  7. Retail Stores 02000-02999

Enter the numbers of the account groups you want to select (e.g., 1-3,5):
```

5. The script will display a progress bar during execution and will notify you once the Terraform files have been successfully created at the specified path. At that point, the Python portion of the process is complete.

Step 4: Check Terraform project

1. Navigate to your terraform project folder.

2. Three files will be generated:

   ○ The first one is the ***providers.tf*** file, which defines the ThousandEyes provider required to initialize the Terraform project, along with the provider blocks for each selected account group. These provider blocks establish the necessary mapping between the imported tests and the account groups they belong to.

   ○ The second file is ***variables.tf***. This file assigns the value of the var.token variable, which is required by providers.tf to authenticate and fetch the data. The token used here should match the same API token that was used by the Python script but it can be replaced.

   ○ The ***imports.tf*** file contains the Terraform import blocks required to bring all existing ThousandEyes tests under Terraform management.

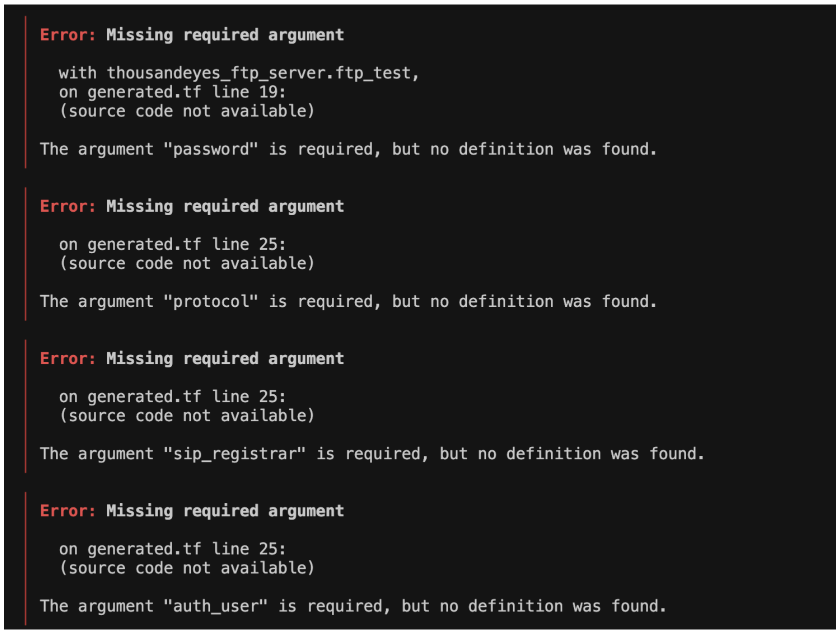
3. Run terraform init to ensure the required providers are downloaded and the project is properly initialized:
   ```
   terraform init
   ```

4. To generate the Terraform state and create a configuration file based on the imported tests, run:
   ```
   terraform plan -generate-config-out=generated.tf
   ```

   Note: Terraform labels this flag as new/experimental, but it is reliable for generating configuration files that match the imported ThousandEyes resources. **You may also replace *generated.tf* with any custom filename you prefer.**

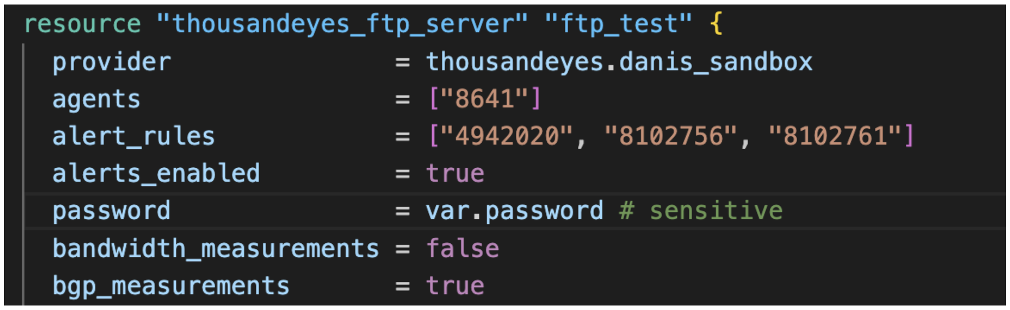
Step 5: Validate and Update Generated Terraform Files

Depending on your configuration, you may encounter conflicts in the ***generated.tf*** file such as the example below. To resolve them, review the errors shown in the console and update the affected Terraform resources with the required values. These values can be provided through variables or (less recommended) hardcoded directly in the resource.



```
Error: Missing required argument

  with thousandeyes_ftp_server.ftp_test,
  on generated.tf line 19:
  (source code not available)

The argument "password" is required, but no definition was found.

Error: Missing required argument

  on generated.tf line 25:
  (source code not available)

The argument "protocol" is required, but no definition was found.

Error: Missing required argument

  on generated.tf line 25:
  (source code not available)

The argument "sip_registrar" is required, but no definition was found.

Error: Missing required argument

  on generated.tf line 25:
  (source code not available)

The argument "auth_user" is required, but no definition was found.
```

For example, in the previous errors, the FTP test resource required a password value. By identifying the corresponding Terraform resource and replacing the missing field with a variable, Terraform will have the necessary input and the validation error will no longer appear. Example:



```
resource "thousandeyes_ftp_server" "ftp_test" {
  provider               = thousandeyes.danis_sandbox
  agents                 = ["8641"]
  alert_rules            = ["4942020", "8102756", "8102761"]
  alerts_enabled         = true
  password               = var.password # sensitive
  bandwidth_measurements = false
  bgp_measurements       = true
```

Note: To verify which parameters are required for each resource, please refer to the official Terraform documentation.
https://registry.terraform.io/providers/thousandeyes/thousandeyes/latest/docs/resources/ftp_server

Step 6: Apply the Configuration and Finalize the Setup

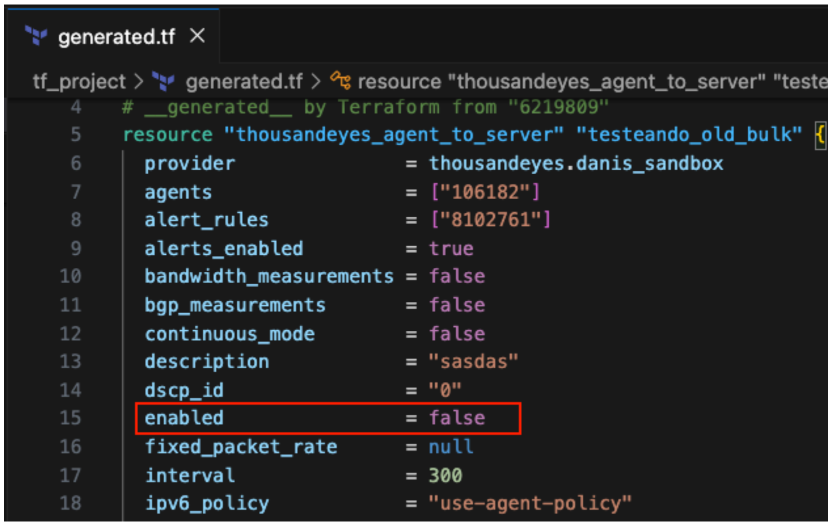
At this point, all configuration issues and validation errors should already be resolved.

1. When you are ready to deploy the configuration, run the following command:
   `terraform apply`

2. Review the execution plan and confirm when prompted.

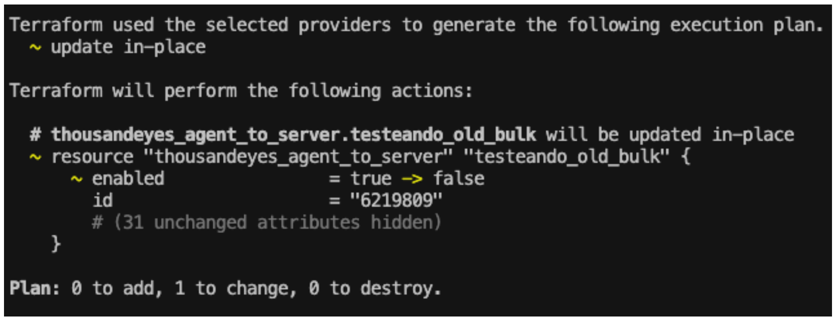
3. Once the apply completes successfully, your ThousandEyes tests will be fully managed through Terraform.

4. From this point forward, you can begin making changes directly in the configuration files (such as generated.tf). Terraform will detect these modifications and apply them during future terraform plan and terraform apply runs.
   Example: You can enable or disable a test by updating the value of the enabled attribute from true to false.


generated.tf
tf_project > generated.tf > resource "thousandeyes_agent_to_server" "teste

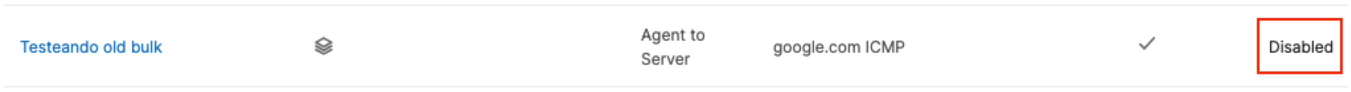
```
# __generated__ by Terraform from "6219809"
resource "thousandeyes_agent_to_server" "testeando_old_bulk" {
  provider                 = thousandeyes.danis_sandbox
  agents                   = ["106182"]
  alert_rules              = ["8102761"]
  alerts_enabled           = true
  bandwidth_measurements   = false
  bgp_measurements         = false
  continuous_mode          = false
  description              = "sasdas"
  dscp_id                  = "0"
  enabled                  = false
  fixed_packet_rate        = null
  interval                 = 300
  ipv6_policy              = "use-agent-policy"
```

5. After making the update, you can verify the changes by running the `terraform plan` command. You should see an output similar to the example below, showing that only one change was detected for the specific test.



```
Terraform used the selected providers to generate the following execution plan.
  ~ update in-place

Terraform will perform the following actions:

  # thousandeyes_agent_to_server.testeando_old_bulk will be updated in-place
  ~ resource "thousandeyes_agent_to_server" "testeando_old_bulk" {
      ~ enabled                    = true -> false
        id                         = "6219809"
        # (31 unchanged attributes hidden)
    }

Plan: 0 to add, 1 to change, 0 to destroy.
```

6. If you are ready to apply the changes, run the terraform apply command again and confirm when prompted.
7. You should now see the updated status reflected for that test at TE platform:



Testeando old bulk		Agent to Server	google.com ICMP	✓	Disabled

(Optional) Helper Scripts

The project includes two optional Python helper scripts designed to provide quick visibility into the ThousandEyes resources available in your account. These scripts retrieve and list information across all account groups, allowing you to easily review agents and tests before running the main automation workflow.

To execute the scripts, run the following commands from the project directory:

```
python helpers/list_agents.py
python helpers/list_tests.py
```

Each script outputs a CSV file containing the retrieved data, formatted using the following headers:

- For agents:
  Account Name, Account ID, Agent Name, Agent ID, Agent Type, Agent Location

- For tests:
  Account Name, Account ID, Test Name, Test ID, Test Type

These scripts are optional but can be useful for auditing, validating resource inventories, understanding the structure of your ThousandEyes environment, or testing Terraform behavior when adding, removing, or modifying agents and tests.

After execution (Python)

1. To check the status of previous API calls, you can open the log file located at:
   `<python project folder>/logs/api_calls.log`

2. If the application encounters an error, you can review the details in:
   `<python project folder>/logs/app.log`

If the script fails, review these log files and share them with the team for further analysis.

Licensing Information

Except as expressly agreed by the parties in writing, your use of the ThousandEyes software is governed by the following terms: https://app.thousandeyes.com/legal/agreement.

In addition, the ThousandEyes software has Apache licensing, you can find the clause under the LICENSE file on the project directory, additionally, the following open-source software licensed under separate terms:

BSD 3-Clause license

Name	Version	License
colorama	0.4.6	BSD 3-Clause license
httpx	0.25.1	BSD 3-Clause license

Note: This license has also been called the “New BSD License” or “Modified BSD License”. See also the 2-clause BSD License.

Copyright

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Service and Support Clause

1. **Scope of Support**
   Technical support will be provided for a period of **3 months** starting from the date of script implementation. This support will cover only issues related to code execution errors attributable to **ThousandEyes** or problems arising from the initial configuration provided by our team.
2. **Exclusions from Support**
   Support does not include:
   - Modifications or customizations requested by the client after the initial implementation.
   - Issues arising from changes in the server environment, operating system updates, or modifications made by unauthorized personnel.
   - Errors caused by misuse of the software or third-party interference.
   - Changes in functional or technical requirements that were not previously agreed upon.
3. **Modifications and Additional Hours**
   If the client requires modifications, enhancements, or adjustments outside the scope of initial support, these will be considered **additional work**. For this, it will be necessary to purchase a package of service hours, which will be billed according to the current rate at the time of the request.
4. **Functionality Guarantee**
   Our team guarantees that the scripts will function according to the technical requirements specified during the initial implementation. However, we are not responsible for issues arising from:
   - Changes in third-party APIs or services.
   - Failures in the client's server infrastructure or network.
   - Unauthorized modifications to the code or server configuration.
5. **Code Ownership**
   The code developed and delivered is the intellectual property of Thousandeyes. The client receives a perpetual usage license for the code but does not have the right to distribute, modify, or resell it without express written authorization.
6. **Limitation of Liability**
   Thousandeyes will not be liable for indirect damages, data loss, or business interruptions resulting from the use of the scripts. The client agrees to use the software at their own risk and is responsible for backing up their data and configurations.
7. **Support Extension**
   Once the 3-month support period has ended, the client may purchase an **extended support plan** to receive ongoing assistance. The terms and costs of this plan will be provided upon the client's request.

Acceptance of Terms

By proceeding with the implementation of the scripts, the client agrees to the terms and conditions outlined in this service clause. Any modifications to these terms must be agreed upon in writing by both parties.