ARL-TR-9679 ● APR 2023

Sensor Information Recommender System Start-Up Guide

by Michael Lee and Tim Gregory

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-9679 ● APR 2023

Sensor Information Recommender System Start-Up Guide

Michael Lee and Tim Gregory
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
April 2023	Technical Report	1/1/2022	9/30/2024

4. TITLE AND SUBTITLE
Sensor Information Recommender System Start-Up Guide

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Michael Lee and Tim Gregory

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
DEVCOM Army Research Laboratory ATTN: FCDD-RLA-IB Adelphi, MD 20783-1138	ARL-TR-9679

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The Sensor Information Recommender System is a proof-of-concept application to test the effectiveness of a Reinforcement Learning algorithm for recommending the right information source that a military analyst could select at that moment. The system has various ways to disseminate the data to the user and provide feedback to the Recommender Server. This report describes a Sensor Information Recommender System comprising data sources, a Recommender Server, a Tactical Assault Kit Server, and Android Team Awareness Kit instances. This report describes the steps to deploy the components in a self-contained environment for testing and development.

15. SUBJECT TERMS
Military Information Sciences, Value of Information, Reinforcement Learning, sensor data processing, ATAK, Cursor-on-Target

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
Unclassified	Unclassified	Unclassified	SAR	23

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Michael Lee	(301) 394-5608

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

1. Introduction

The Sensor Information Recommender System is a software system that determines and disseminates high-value information to an end user. The main components of the system are the Recommender Server, Tactical Assault Kit (TAK) Server, Android Team Awareness Kit (ATAK), and ATAK plug-in. As a fielded system, each component would ideally run independently on its own dispersed instance. However, during development, it is helpful to run each part of the system in the same platform to quickly test the data flow. It is also helpful to run all the components in a self-contained environment to demonstrate the Value-of-Information (VOI) System to an audience. This report describes the steps to deploy these components in a self-contained environment for testing and development.

2. System Components

The system architecture is shown in Fig. 1. The Recommender Server receives data from the deployed sensors and sends the recommended information objects from the sensors to ATAK via the TAK server.

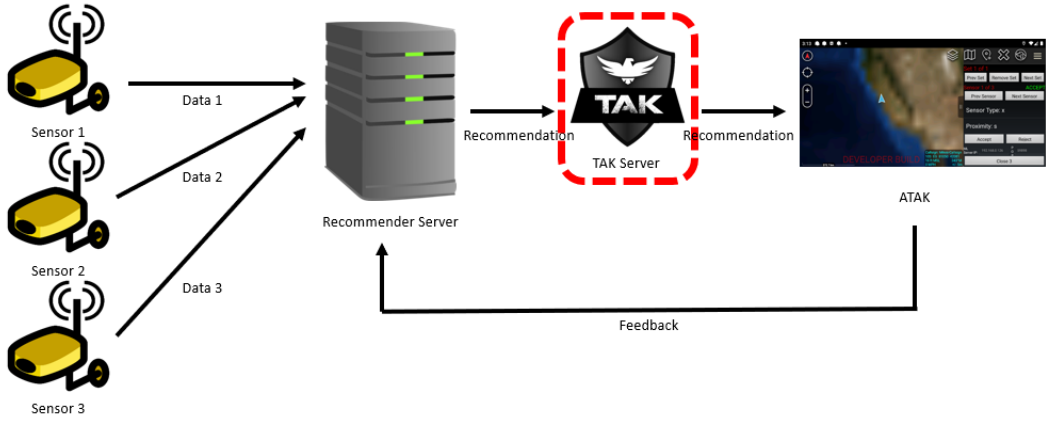

Fig. 1 TAK Server in the Sensor Information Recommender System workflow

3. TAK Server (in Docker Container)

The TAK Server* acts as a mediator between the Recommender Server and one (or many) ATAK instances. It receives recommendations from the Recommender Server and transmits them to the ATAK instances.

* To download the TAK Server, visit www.tak.gov.

3.1 Starting the Docker Image

There are many ways to prepare an instance of a TAK Server:

- Full installation on a Linux server
- Virtual machine
- Docker image

This section describes the process of starting a Docker image of the TAK Server on a Windows OS:

1) Run the Docker Desktop application and start the Docker service (Fig. 2).

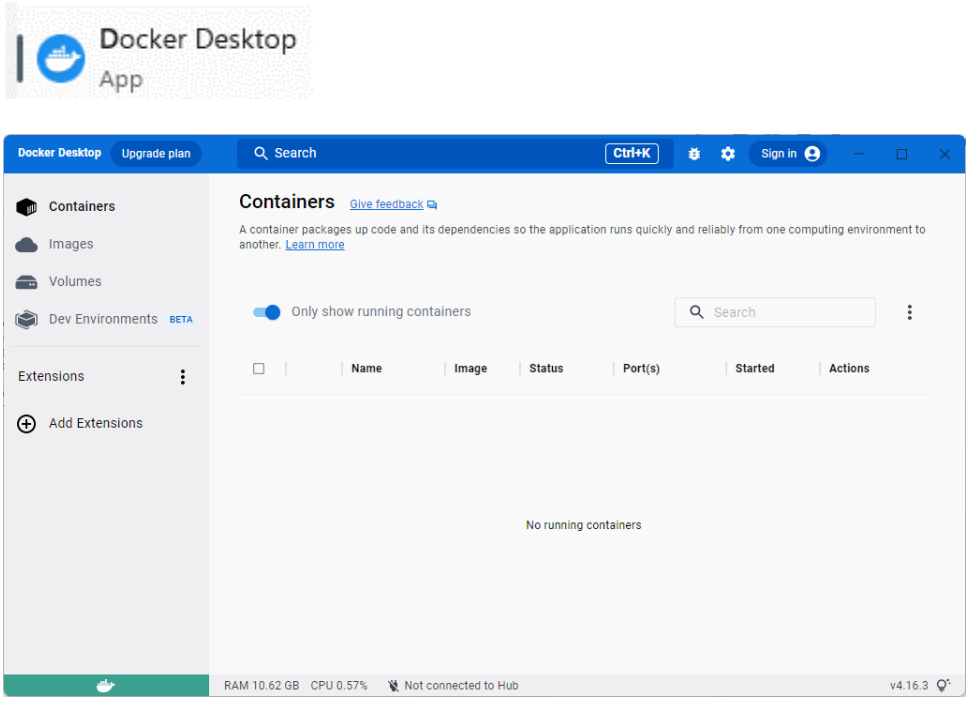

Fig. 2 Docker Desktop application on Windows OS

2) Start a Windows Subsystem for Linux (WSL) terminal application (e.g., "Ubuntu on Windows") (Fig. 3). Change the directory to the path where the TAK Server container was installed (e.g., "/home/mike/TAK/tak-server/").

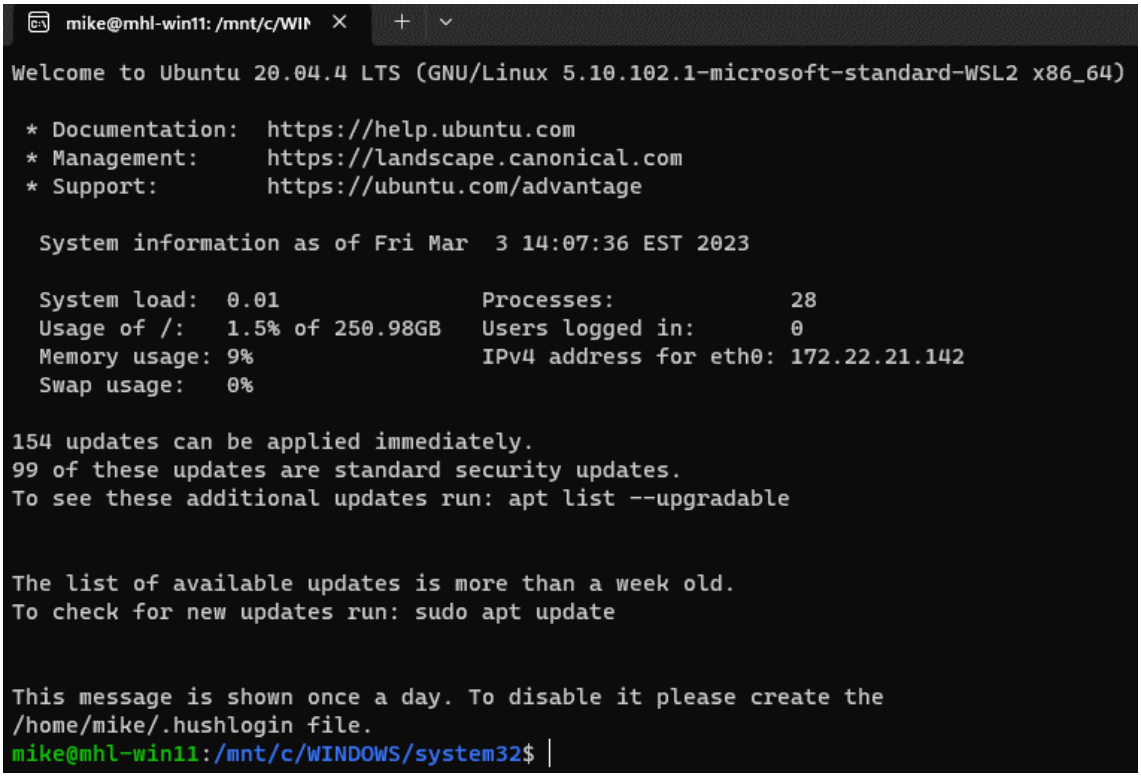

Fig. 3 Windows Subsystem for Linux (WSL) terminal

3) Start the TAK Server–related containers with the "docker-compose up" command (Fig. 4). This command will occupy the terminal and print the TAK Server status messages. To run the TAK Server in daemon mode in the background, add the "-d" argument (e.g., "docker-compose up -d").

```
mike@mhl-win11:~/TAK/tak-server$ docker-compose up
[+] Running 4/3
 ⠿ Network tak-server-raven-plugin-training_default  Created       0.0s
 ⠿ Container takserver-db                            Created       0.1s
 ⠿ Container takserver                               Created       0.1s
 ⠿ Container raven-simulator                         Created       0.1s
Attaching to raven-simulator, takserver, takserver-db
takserver-db     | The files belonging to this database system will be owned
by user "postgres".
takserver-db     | This user must also own the server process.
takserver-db     |
takserver-db     | The database cluster will be initialized with locale "C".
takserver-db     | The default database encoding has accordingly been set to
"SQL_ASCII".
takserver-db     | The default text search configuration will be set to
"english".
takserver-db     |
takserver-db     | Data page checksums are disabled.
takserver-db     |
takserver-db     | fixing permissions on existing directory
/var/lib/postgresql/data ... ok
takserver-db     | creating subdirectories ... ok
takserver-db     | selecting default max_connections ... 100
takserver-db     | selecting default shared_buffers ... 128MB
takserver-db     | selecting default timezone ... Etc/UTC
takserver-db     | selecting dynamic shared memory implementation ... posix
takserver-db     | creating configuration files ... ok
takserver-db     | running bootstrap script ... ok
takserver-db     | performing post-bootstrap initialization ... ok

. . .

takserver-db     | 19:18:39.930 [main] INFO  com.bbn.tak.schema.UpgradeCommand
- Successfully applied 39 update(s).
takserver-db     | 19:18:39.931 [main] INFO  com.bbn.tak.schema.UpgradeCommand
- TAK server database schema is up to date.
takserver-db     | Database updated with SchemaManager.jar
takserver-db     | Systemctl was not found. Skipping Systemd configuration.
takserver-db     | 19:18:40.498 [main] INFO  com.bbn.tak.schema.SchemaManager
- trying to load configuration from file CoreConfig.xml
takserver-db     | 19:18:40.501 [main] INFO  com.bbn.tak.schema.SchemaManager
- trying to load CoreConfig.xml in /opt/tak
takserver-db     | 19:18:41.207 [main] INFO
com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Starting...
takserver-db     | 19:18:41.402 [main] INFO
com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Start completed.
takserver-db     | 19:18:41.459 [main] INFO  com.bbn.tak.schema.UpgradeCommand
- Database is not empty.
takserver-db     | 19:18:41.675 [main] INFO  com.bbn.tak.schema.UpgradeCommand
- TAK server database schema is up to date.
```

Fig. 4 Status messages printed from the “docker-compose up” command

4) Open a browser to “http://localhost:8080/index.html” to access the TAK Server administration page. Use the ID and password created during the installation to log in to the administration page (Fig. 5).

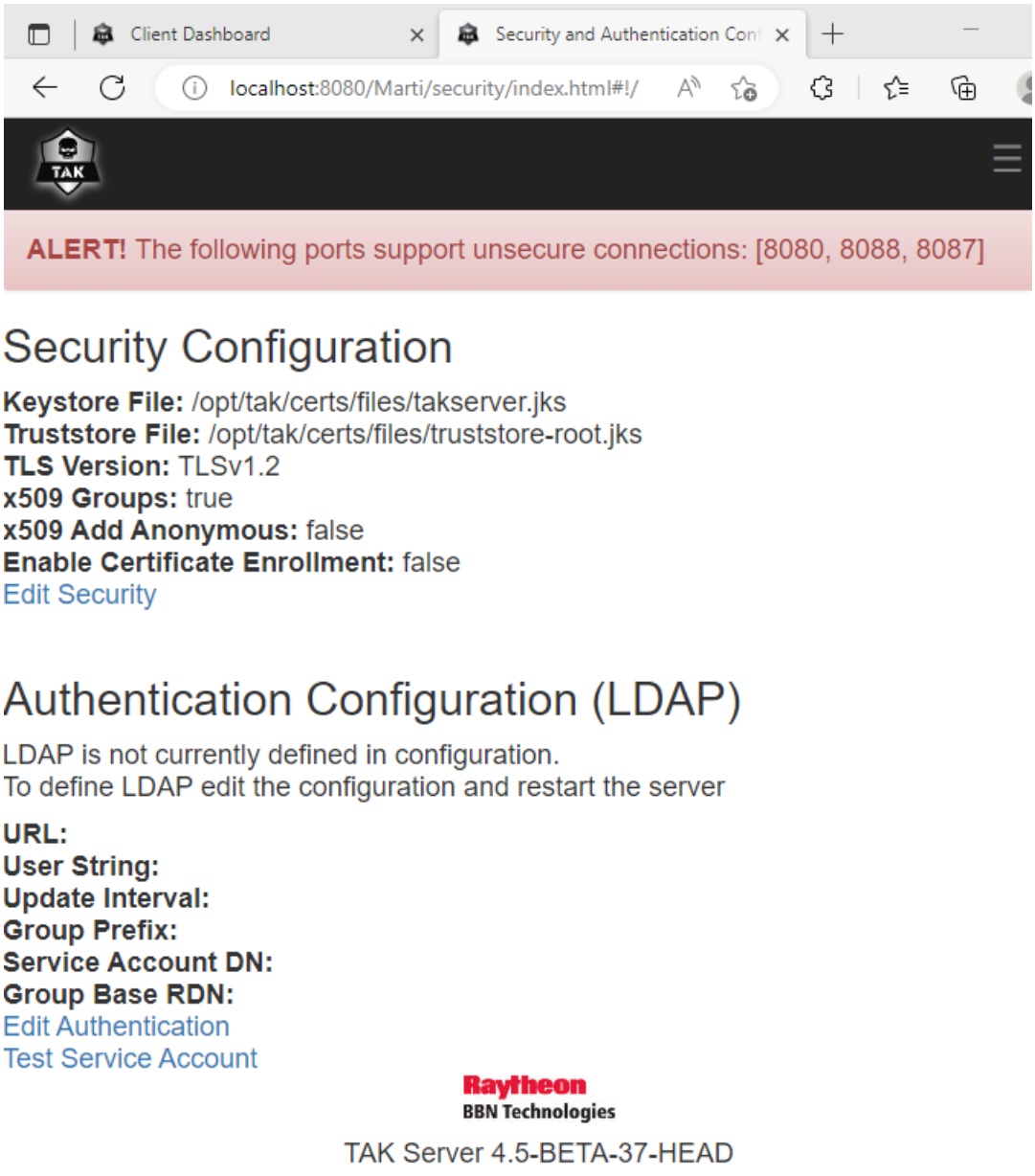

Fig. 5 TAK Server administration web application

5) View the Docker Desktop application and verify that the Docker containers are now running (Fig. 6).

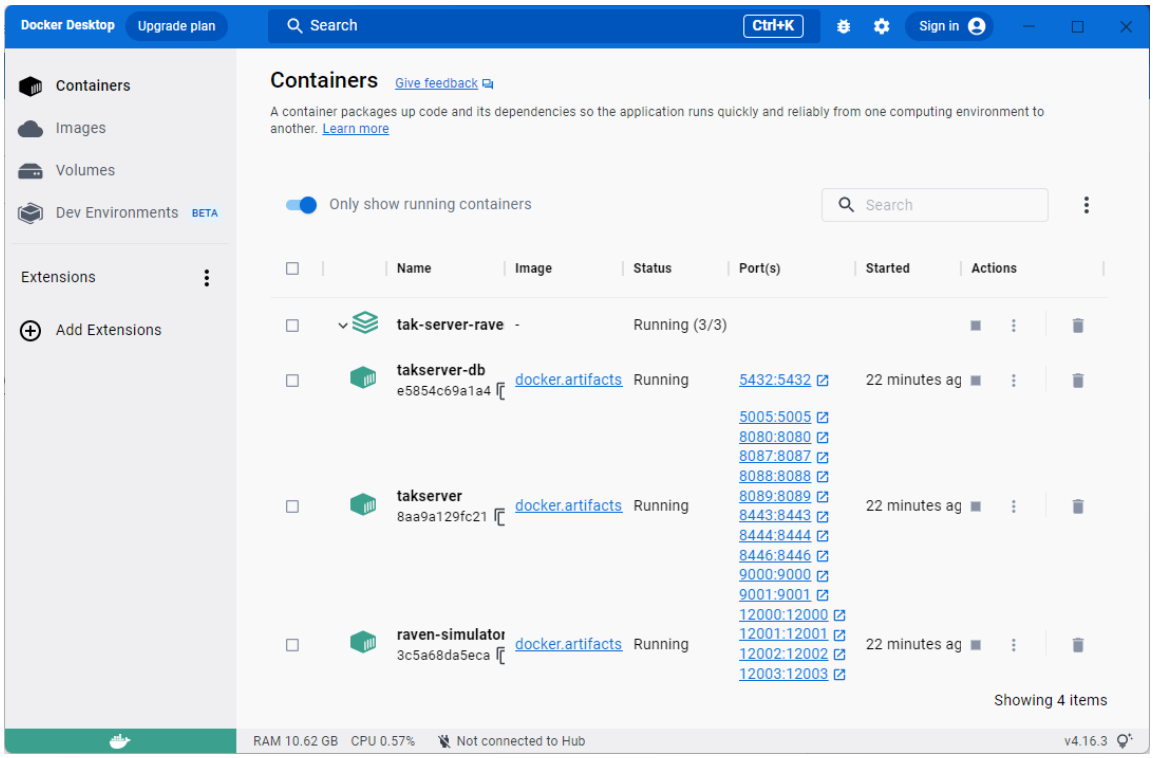

Fig. 6 TAK Server containers running in Docker

3.2 Shutdown Command

A dedicated shutdown command must be executed to properly stop the TAK Server:

1) Start a new WSL terminal application. Change the directory to the path where the TAK Server container was installed (e.g., "/home/mike/TAK/tak-server/").

2) Stop the running TAK Server–related containers with the "docker-compose down" command (Fig. 7).

```
mike@mhl-win11:~/TAK/tak-server$ docker-compose down
[+] Running 4/4
 ⠿ Container raven-simulator                          Removed    1.7s
 ⠿ Container takserver                                Removed   11.0s
 ⠿ Container takserver-db                             Removed   10.6s
 ⠿ Network tak-server-raven-plugin-training_default   Removed    0.2s
mike@mhl-win11:~/TAK/tak-server$
```

Fig. 7 Status messages printed from the "docker-compose down" command

4. ATAK (in Android Studio Emulator)

ATAK* receives the model recommendations from the TAK Server and allows the users to provide feedback about the sensor data. The user feedback will be forwarded to the Recommender Server. Figure 8 displays the ATAK workflow.

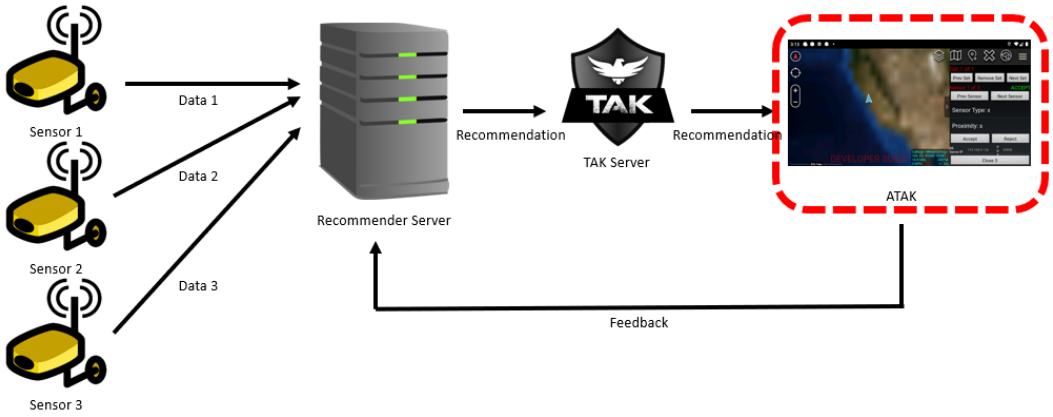

Fig. 8 ATAK in the Sensor Information Recommender System workflow

4.1 Starting ATAK in Android Studio's Emulator

ATAK can be deployed to a physical Android tablet or run in an emulator. This section describes the process of starting ATAK in Android Studio's emulator.

1) Start Android Studio. Click on the "Device Manager" button and launch any emulator of choice (Fig. 9).

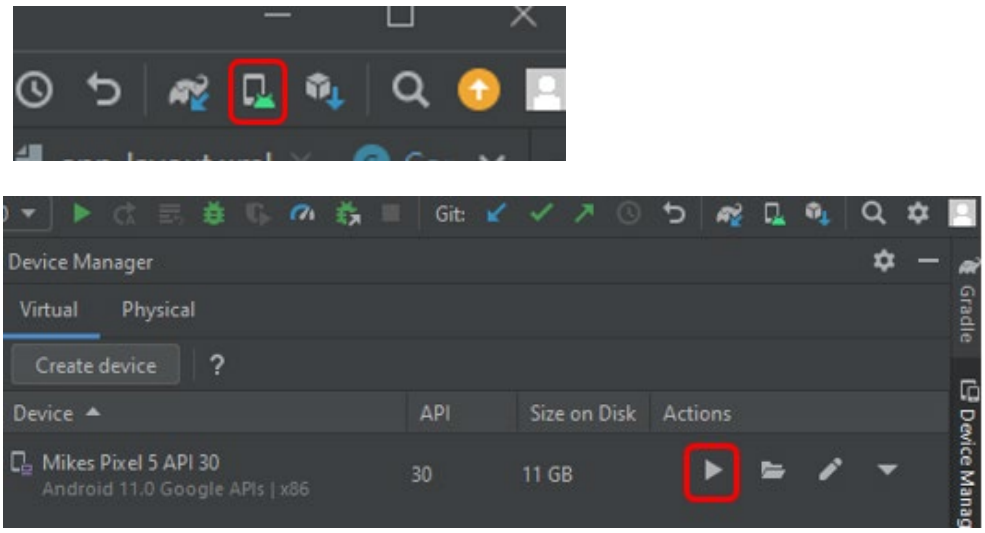

Fig. 9 Android Studio's Device Manager

*Android Team Awareness Kit (ATAK), Ver. 4.3.0.3. TAK Product Center. https://tak.gov/products.

2) Start the ATAK application in the emulator (Fig. 10). When the ATAK application loads, it will display a marker on a map (Fig. 11).

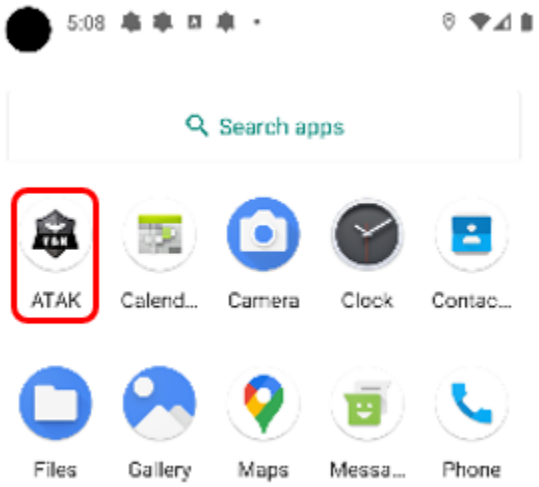

Fig. 10 ATAK application logo in Android

Fig. 11 ATAK initial screen

4.2 Connecting to the TAK Server

ATAK must connect to the TAK Server to subscribe to the data published by the TAK Server.

1) In ATAK, open the "Add Streaming CoT Endpoint" window by navigating to:

   Settings -> "Network Preferences" -> "Server Connections" -> "…" button -> Add

Create a connection to the TAK Server by entering the following values on the form (also see Fig. 12). The IP should be the same IP of the server where the TAK Server container is running.

Name:	TAKServer_RavenDocker
Address:	192.168.0.126
Advanced Options:	Checked
Streaming Protocol:	TCP
Server Port:	8088
Use default SSL/TLS Certificates:	Checked

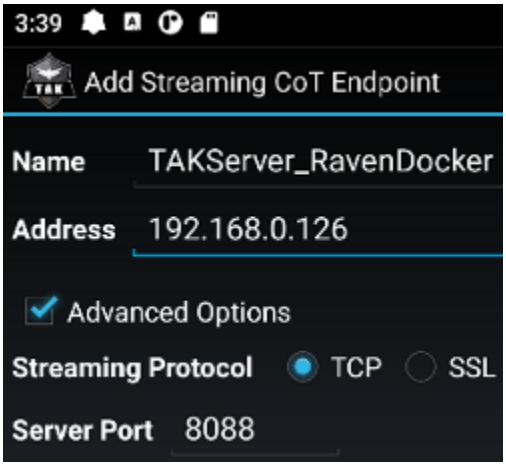

Fig. 12 ATAK configuration window to connect to a TAK Server

2) If all the entered values are valid, a new entry will appear in the list of the server connections (Fig. 13).

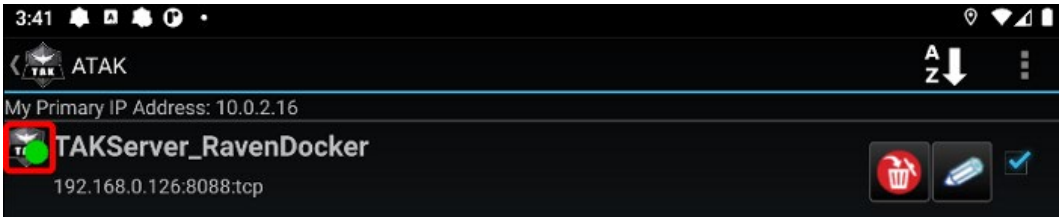

Fig. 13 ATAK lists a connection to the TAK Server

The green dot and the checkbox mean that ATAK is connected to the TAK Server. To disconnect from the TAK Server, uncheck the box and the green dot will turn red (Fig. 14).

Fig. 14 Red dot indicates ATAK is disconnected from the TAK Server

3) Verify ATAK is connected to the TAK Server.

Open a browser to the TAK Server's admin page (e.g., http://localhost:8080). Navigate to:

Menu -> Monitoring -> "Client Dashboard"

This page lists all clients that are currently connected to the TAK Server (Fig. 15).

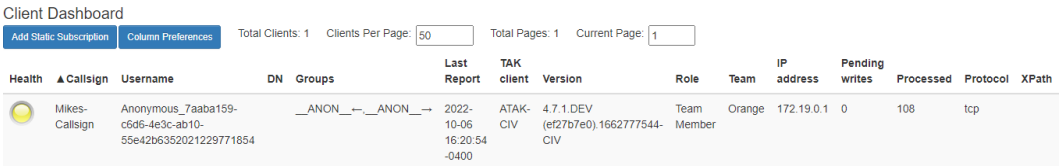

Fig. 15 TAK Server's admin page lists the connected clients

5. Recommender Server

The Recommender Server is the primary component of the Sensor Information Recommender System (Fig. 16). It receives the sensor data from the deployed sensors and manages the trained model. Initial recommendations made by the model will be forwarded to the TAK Server.

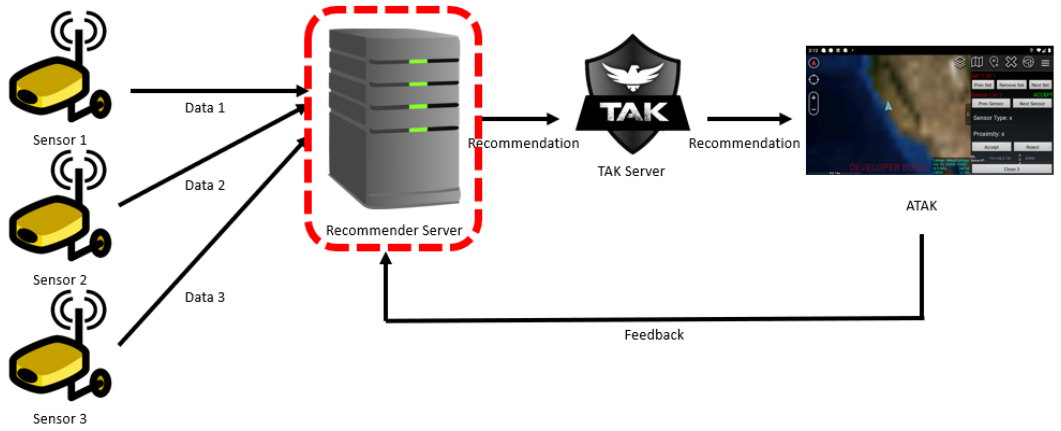

Fig. 16 The Recommender Server in the Sensor Information Recommender System workflow

The Recommender Server is a single Python script. Before running the Recommender Server, several properties must be set.

5.1 Configuration

Reinforcement Learning Properties:

There are many properties that affect the Reinforcement Learning algorithm in the Recommender Server's Python script. Most of the properties can be ignored unless there is a specific need to adjust the inner workings of the algorithm. Two high-level properties of the Reinforcement Learning algorithm that may change often are the numeric values of "Number of Arms" in the Multi-Armed Bandit method and "Top-K" to indicate the number of recommended objects sent to the user from the top-K ranking of mostly likely to be chosen by the user. The Number of Arms property is essentially the number of information objects to choose from. For example, to configure the algorithm to run with five arms and to choose Top-3 results, edit the Python file as follows:

```
choose_k    = 3
arms        = 5
```

Socket Server Properties:

The Recommender Server runs a Socket Server on a dedicated thread that listens for incoming messages. These incoming messages include data from the deployed sensors and feedback data from ATAK users. The server's IP address and port number are required by the deployed sensors and ATAK to establish a connection to the Recommender Server. While the server's IP is not specified in the Python script, the Socket Server port is defined in the Python file. For example, to configure the Socket Server to listen on port 59898, edit the Python file as follows:

```
port  = 59898
```

Socket Client Properties:

The Recommender Server also acts as a client when it needs to distribute Top-K model recommendations to ATAK instances. In this workflow, the TAK Server manages communication with all the ATAK instances. To establish a connection to the TAK Server, the Recommender Server needs to know the IP address and port number of the TAK Server. For example, to define "192.168.0.126" and "8088" as the TAK Server's address, edit the Python file as follows:

```
_PORT_OfTAKServer = 8088
_IP_OfTAKServer   = "192.168.0.126"
```

5.2 Running the Server

After the server properties have been updated, running the Recommender Server is similar to running any other Python script.

1) Change the directory to where the Recommender Server Python file "sensorVOI.py" is located.

```
C:\> cd C:\Workspace\RL-VOI
```

2) Activate the Conda environment for the Recommender Server.

```
C:\Workspace\RL-VOI> conda activate RL-VOI
(RL-VOI) C:\Workspace\RL-VOI>
```

3) Run the Recommender Server Python file. This terminal window will be used by the Recommender Server to print status messages while in operation.

```
(RL-VOI) C:\Workspace\RL-VOI> python sensorVOI.py
```

5.3 Shutting Down the Server

The Recommender Server can be shut down by running a separate Python script called "sensorVOI_toKillSocketServer.py". (Note that pressing "Control + c" will not shut down the server.) The "sensorVOI_toKillSocketServer.py" script sends a kill command to the server to instruct the server to perform the shutdown sequence. The "sensorVOI_toKillSocketServer.py" needs to be edited to define the IP address and port number of the Recommender Server. For example, to define that the Recommender Server is listening for incoming messages on port 59898, edit the Python file as follows:

```
ipOfServer      = socket.gethostname()
portOfServer    = 59898
```

After defining the IP address and port number, run the Python script to shut down the Recommender Server.

```
(RL-VOI) C:\Workspace\RL-VOI> python
sensorVOI_toKillSocketServer.py

Connected to Server (win11:59898).
Sent msg 'KILL_SERVER_THREAD'
Closed connected to Server (win11:59898)...
-----------------------------------
```

6. Simulating Deployed Sensors

The Sensor Information Recommender System receives data from deployed sensors. However, during development and testing, a separate program can be used to simulate the sensors and transmit data to the Recommender Server. The simulation program will allow the developers to test or demonstrate the system to an audience (Fig. 17).

A Python script called "clientSimulator_SendsDataToServer.py" is included with the Sensor Information Recommender System. This simulates a sensor that generates a line of 23 comma-separated values (CSVs) string enclosed in a Cursor-on-Target (CoT) message format. Before running the Simulation script, several properties must be defined. The following subsection describes how to simulate sensor data output using a CSV file.

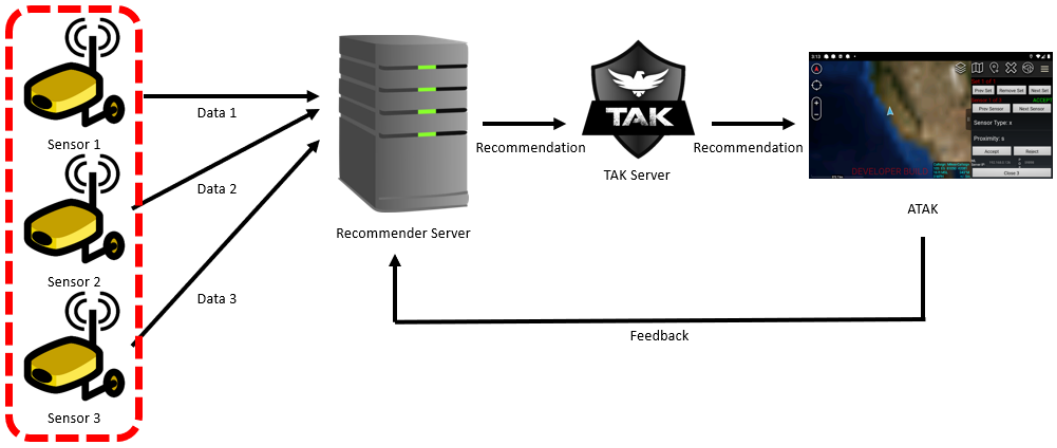

Fig. 17 Simulation program in the Sensor Information Recommender System workflow

6.1 Configuration

Socket Client Properties:

The Simulation program is a client that transmits data to a Socket Server (i.e., Recommender Server). To establish a connection to the Recommender Server, the Simulation program needs to know the IP address and port number of the server. For example, to define "192.168.0.126" and "59898" as the Recommender Server's address, edit the Python file as follows:

```
ipOfServer      = "192.168.0.126"
portOfServer    = 59898
```

Data Source Properties:

In this simulation, the open-source Mushroom* data set from the UCI Machine Learning Repository was used. The Mushroom data set contains 23 CSVs representing the features for 8,124 mushrooms. The data sent by the Simulation program references the open-source Mushroom data set. After downloading the Mushroom CSV file, the CSV file in the Python file is referenced as follows:

```
pathToDataCSVFile = "..\\mushroom\\agaricus-
lepiota.data.csv"
```

CoT Data Properties:

The Simulation program wraps the Mushroom CSV data in a CoT message format. In order to generate a CoT message, the CoT message properties (e.g., event ID, latitude, longitude) must be populated for each CoT message. At the moment, the Simulation program is not able to randomly generate CoT message properties to produce a specified number of CoT messages wanted by the user. As a result, the user must specify the CoT message properties for each predefined number of CoT messages to be sent by the Simulation program. However, the actual Mushroom CSV is automatically populated from the Mushroom CSV file. For example, in order to configure the Simulation program to send six CoT messages to the Recommender Server 5-s apart, edit the Python file as shown in Fig. 18:

```
numSecsBetweenEvents = 5

msgsToSend = [
    ["COT_W_VOI_MUSHROOM", "ML_8000a",  "b-i-e;mushroomdata",
"m-g",    39.028725, -76.963787,  0.0,    -1,
"[GET_FROM_DATA_FACTORY]" ],
    ["COT_W_VOI_MUSHROOM", "ML_8000b",  "b-i-e;mushroomdata",
"m-g",    39.028725, -76.963787,  0.0,    -1,
"[GET_FROM_DATA_FACTORY]" ],
    ["COT_W_VOI_MUSHROOM", "ML_8000c",  "b-i-e;mushroomdata",
"m-g",    39.028725, -76.963787,  0.0,    -1,
"[GET_FROM_DATA_FACTORY]" ],
    ["COT_W_VOI_MUSHROOM", "ML_8000d",  "b-i-e;mushroomdata",
"m-g",    39.028725, -76.963787,  0.0,    -1,
"[GET_FROM_DATA_FACTORY]" ],
    ["COT_W_VOI_MUSHROOM", "ML_8000e",  "b-i-e;mushroomdata",
"m-g",    39.028725, -76.963787,  0.0,    -1,
"[GET_FROM_DATA_FACTORY]" ],
    ["COT_W_VOI_MUSHROOM", "ML_8000f",  "b-i-e;mushroomdata",
"m-g",    39.028725, -76.963787,  0.0,    -1,
"[GET_FROM_DATA_FACTORY]" ],
]
```

Fig. 18 CoT properties for the Simulation program

* To download the data set, visit https://archive.ics.uci.edu/ml/datasets/Mushroom.

6.2 Running the Simulator

After the server properties have been updated, running the Simulation program is similar to running any other Python script.

1) Change the directory to where the Recommender Server Python file is located.

```
C:\> cd C:\Workspace\RL-VOI
```

2) Activate the Conda environment for the Recommender Server.

```
C:\Workspace\RL-VOI> conda activate RL-VOI
(RL-VOI) C:\Workspace\RL-VOI>
```

3) Run the Simulation program Python file.

```
(RL-VOI) C:\Workspace\RL-VOI> python
clientSimulator_SendsDataToServer.py
```

7. Conclusion

The Sensor VOI application is a proof-of-concept system to leverage reinforcement learning technology to help warfighters quickly access the right information they need at that moment. During the development process, the entire system workflow can be deployed on a single self-enclosed environment to facilitate debugging and testing. This report describes the component initialization sequence and how to configure the components.

List of Symbols, Abbreviations, and Acronyms

ATAK	Android Team Awareness Kit
CoT	Cursor-on-Target
CSV	comma-separated value
ID	identification
IP	Internet Protocol
OS	operating system
TAK	Tactical Assault Kit
VOI	Value of Information
WSL	Windows Subsystem for Linux

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CI TECH LIB
2 (PDF)	DEVCOM ARL FCDD RLA IB M LEE T GREGORY