



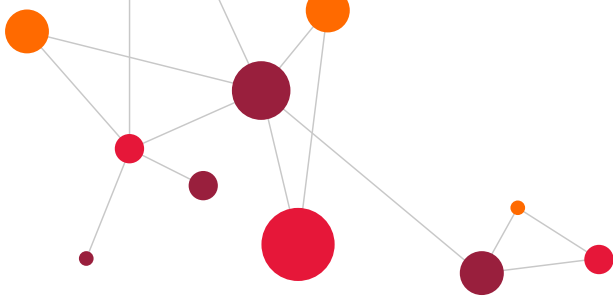
Intelligent Ship Artificial Intelligence Network (ISAIN)

Progeny Task 25: WP 2.1

Admin Guide

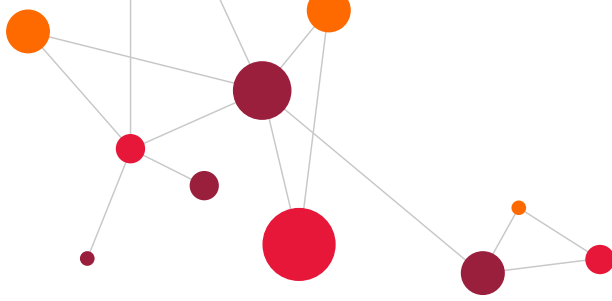
Issue: 2.2

Document Reference: SSL/11145/DOC/0006



Conditions of Supply - Full Rights

The document is supplied to MOD as a FULL RIGHTS VERSION under the terms of DEFCON 703 (Edn 11/02), with ownership of the outputs herein vested in the Authority.



Authorisation

Role	Name(s)
Author	Jacob Pennels, Ian Brandon, Ross Walker
Reviewed	Carl Froom
Authorised for Release	David Wales

Distribution

Copy Number(s)	Recipient
1	Dstl
2	CGI

Revision History

Version	Date	Author	Description
0.1	09/03/2020	Jacob Pennels	Initial draft
0.2	29/05/2020	Carl Froom	Minor updates
0.3	01/06/2020	Jacob Pennels	Updated to reflect latest functionality
0.4	04/06/2020	Carl Froom	Updated during review
0.5	05/06/2020	Danny Saggo	Updated during review
1.0	05/06/2020	Carl Froom	Initial Release
2.0	08/02/2022	Ross Walker	Updated to reflect status at delivery of Phase 2
2.1	05/04/2022	Ross Walker	Updated installation procedure for ELK
2.2	13/06/2022	Carl Froom	Replaced references to IDDC goal decomposer with IDDC Dataflow Inferer

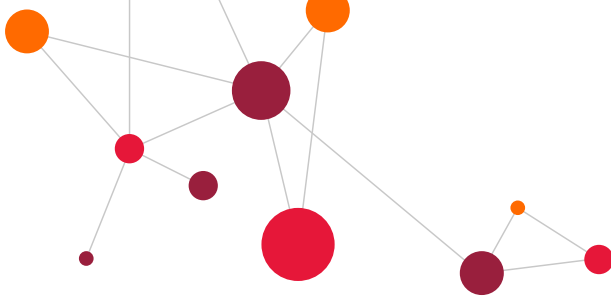
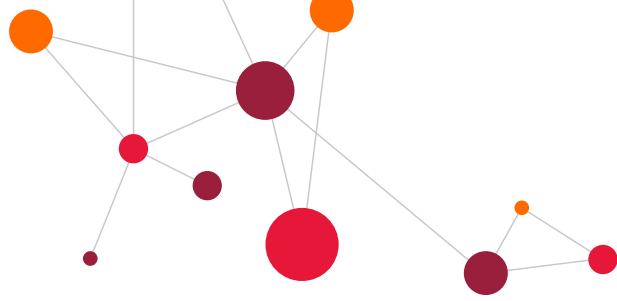
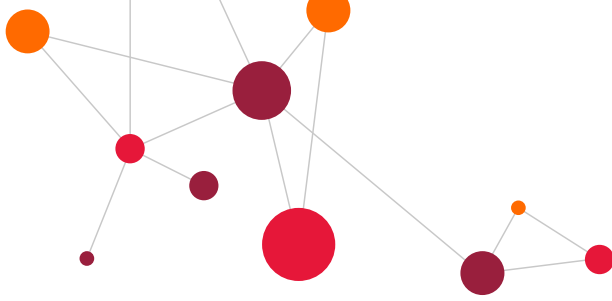


Table of contents

1	Introduction	1
1.1	BACKGROUND	1
1.2	PURPOSE	1
1.3	REFERENCES	1
2	Overview	2
3	Software Stack	3
4	Installation	4
5	ApacheDS	6
5.1	OVERVIEW	6
5.2	DEFAULT USERS & GROUPS	6
5.3	USER CONFIGURATION	7
5.3.1	LDIF FILE	7
5.3.2	MANUAL CONFIGURATION	9
6	Console	11
6.1	OVERVIEW	11
6.2	INITIAL CONFIGURATION	11
6.3	CONFIGURING TABS	13
7	Apache NiFi	16
7.1	OVERVIEW	16
7.2	POLICY MANAGEMENT	16
7.3	VARIABLE CONFIGURATION	16
7.4	NETWORKED (HOOTL) AND NON-NETWORKED (HITL) CONFIGURATION	17
7.5	SIMULATION ENVIRONMENT	17
8	Elastic Stack	18
8.1	ADDING A NEW COMPONENT	18
9	ISAIN Dynamic Dataflow Configuration (IDDC)	20
10	AI Integration	22
10.1	OVERVIEW	22
10.2	AI INTEGRATION	22
10.2.1	AI INPUT/OUTPUT FORMAT	22
10.2.2	AI COMMUNICATION METHODS	23
10.2.2.1	DDS	24
11	Real Time Troubleshooting	26
11.1	OVERVIEW	26
11.2	NIFI	26
11.3	KIBANA	28
11.4	RESETTING THE NETWORK	29



12	Post Scenario analysis	31
	Appendix A - Glossary	33
	Appendix B – Example LDIF File	34



1 Introduction

1.1 Background

The Defence Science and Technology Laboratory (DSTL) has embarked on an Intelligent Ship programme, which will revolutionise ship design by harnessing automation and artificial intelligence to transform naval doctrine.

The Intelligent Ship Artificial Intelligence Network (ISAIN) will provide DSTL with a framework to support a programme of experimentation with Artificial Intelligence (AI) collaboration and human-machine teaming. This will act as a 'playground' for AIs: a 'sandpit' to support inter-relationships between applications and human users, with the focus on demonstrating innovative, challenging and revolutionary concepts and opportunities

1.2 Purpose

This document describes the process of installing, configuring, monitoring and troubleshooting the ISAIN application, and is aimed at the administrator of the system. The guide will cover all aspects of the application and includes all steps that need to be taken to ensure the application runs correctly and as expected.

1.3 References

ID	Reference	Title	Version	Date
1	SSL/11145/DOC/0004	ISAIN Developer Guide	4.0	08/02/2021
2		AdLink OpenSplice Community Edition (https://github.com/ADLINK-IST/opensplice)	N/A	undated

Table 1: References

2 Overview

ISAIN provides a network across which disparate ship systems and AIs can communicate and coordinate alongside human operators, who monitor and make decisions when required. Figure 2 shows the layout of the network. This guide explains the installation process for the entire network, as well as how to configure the NiFi flow, the console, the LDAP server and users and how to use the Kibana UI, to analyse and understand the logs produced by the network at all points.

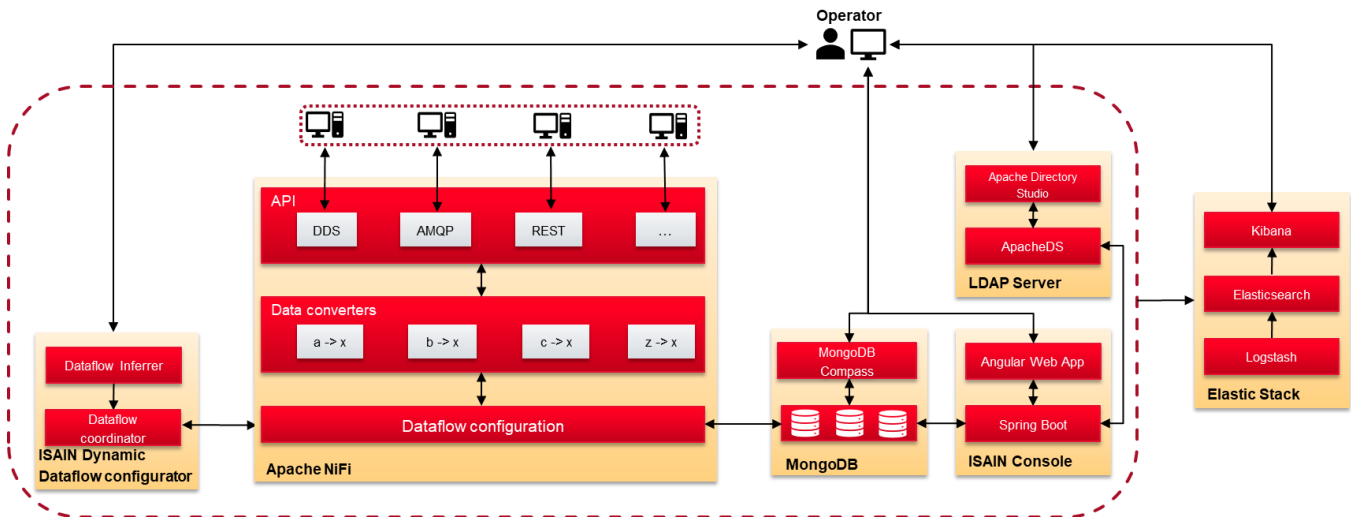
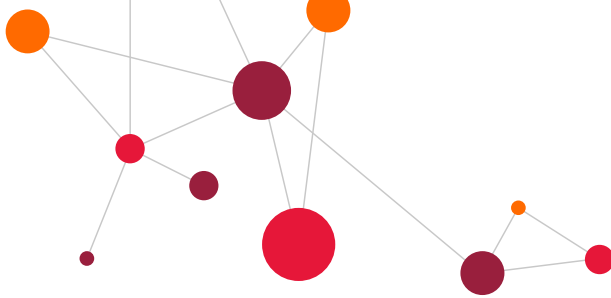


Figure 1: Architecture overview

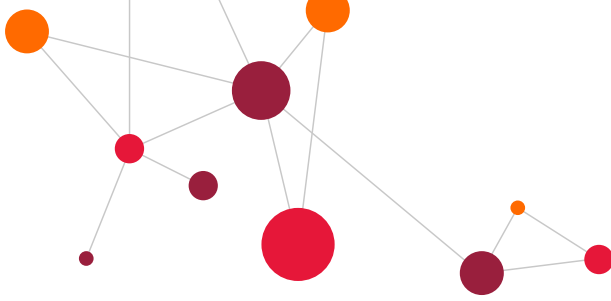


3 Software Stack

ISAIN is built upon the 3rd party applications listed in Table 2.

Name	Version	URLs
Apache NiFi	1.11.2	https://nifi.apache.org https://nifi.apache.org/docs/html
Apache Zookeeper	3.5.6	https://zookeeper.apache.org
ApacheDS	2.0.0.AM25	https://directory.apache.org
Apache Directory Studio	2.0.0.v20180908-M14	https://directory.apache.org/studio
MongoDB	4.2.3	https://www.mongodb.com
MongoDB Compass Community	1.20.5	https://www.mongodb.com/products/compass
Docker	18.09.7	https://www.docker.com
Elastic Stack (Elasticsearch, Logstash, Kibana and Filebeat)	7.6.0	https://www.elastic.co https://www.elastic.co/guide
Fast Downward	20.06	https://www.fast-downward.org

Table 2: Software stack



4 Installation

ISAIN consists of a number of Docker containers and has been tested on Centos 7.6.1810 and Ubuntu 18.04. Through the use of Docker, ISAIN should also be able to run on Windows 10 and Windows Server 2016, however this has not been tested and as such, this guide assumes a Centos or Ubuntu installation

To install and run ISAIN, follow these steps:

1. Install Docker (www.docker.com) and Docker Compose (docs.docker.com/compose) either by installing from the Centos/Ubuntu package repositories or by downloading from www.docker.com
2. Ensure that user namespaces have not been enabled in the Docker configuration file `/etc/docker/daemon.json` ISAIN uses host networking which is incompatible with user namespaces
3. Load each of the Docker images. These are supplied on the USB **ISAIN_2_DEL**, within `docker-images.zip`. There are 17 files to load, which can be done using a single command inside the 'docker-images' folder:

```
for a in `ls`; do docker load -i $a; done;
```

4. On USB extract `deployment-ISAIN-ELK.zip` into `/opt` and run these commands to correctly set file permissions:

- `sudo cd /opt`
- `sudo tar xvf deployment-ISAIN-ELK.tar.gz`
- `sudo chmod -R 777 /opt/isain`
- `sudo chown -R [NON-ROOT-USER]:[NON-ROOT-GROUP] /opt/isain`

5. Move ELK to a separate directory:

- `sudo mkdir /opt/elk`
- `sudo cd /opt/isain/elk`
- `sudo mv elasticsearch/ kibana/ logstash/ nginx/ docker-compose.yml /opt/elk/`

If ELK is on a separate server, move `/opt/elk` there

- `sudo chmod -R 777 /opt/elk`
- `sudo chown -R [NON-ROOT-USER]:[NON-ROOT-GROUP] /opt/elk`

6. Ensure Nginx 'proxy_pass' IP addresses correspond to the server they are installed on:

- `/opt/isain/nginx/conf/default.conf`

7. Navigate to `/opt/isain` and run:

- `docker-compose up -d`

This will start the full ISAIN stack in detached mode.

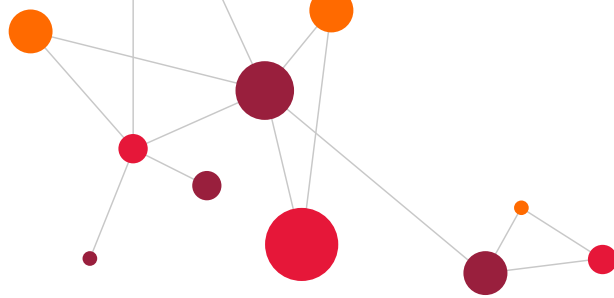
8. Set-up Kibana by insuring all IP addresses correspond to the correct server hosting ELK in:

- `/opt/elk/kibana/config/kibana.yml`
- `/opt/elk/nginx/conf/default.conf`

9. Navigate to `/opt/elk` and run:

- `docker-compose up -d`

This will start ELK in detached mode.

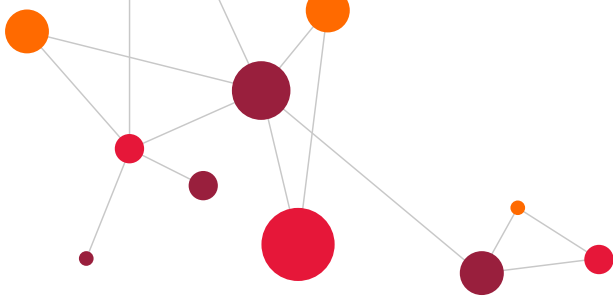


10. Open Kibana in web-browser at <http://localhost:5061/>

- If a 'first-time' pop-up displays click "explore on my own"
- Click [≡] symbol in top-left to show menu
- Navigate to "Analytics" → "Canvas"
- Drag "canvas.json" file onto screen (USB: ISAIN/Kibana/canvas.json)
- Click [≡] symbol in top-left to show menu
- Navigate to "Management" → "Stack Management"
- In new menu navigate to "Kibana" → "Saved Objects"
- On right-side import "dashboard.ndjson" (USB: ISAIN/Kibana/dashboard.json)

11. Kibana Dashboard and Canvas available in [≡] menu under "Analytics"

The containers use the host's network when running, and this can lead to possible conflicts with other applications. When first starting the network, monitor the logs for any of these errors, and consult the Docker Compose documentation for how to solve any conflicts. The containers also use their own hostname, 'isain-svr', which can be used to reference other containers. This keeps the network contained and stops interference with outside applications, and it is best practise to use this value if possible when configuring.



5 ApacheDS

5.1 Overview

LDAP is a commonly used protocol for authentication and authorisation and is used within ISAIN. By default ISAIN uses ApacheDS at its LDAP server.

5.2 Default Users & Groups

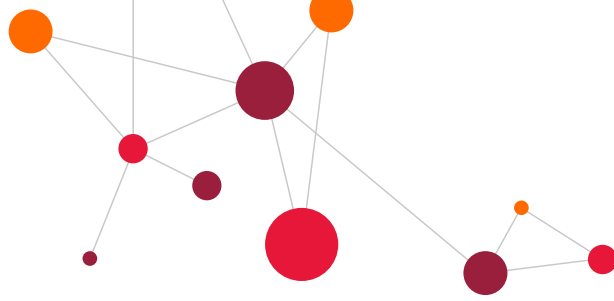
The ApacheDS Docker image comes supplied with a number of default users and groups that allow instant access to the network. Table 3 shows the default users, and Table 4 shows the default groups. Note that some groups contain whole groups as members, meaning all the users in that group are also included in the new group.

USERNAME	PASSWORD
CO1	CO1password
AWO1	AWO1password
AWO2	AWO2password
PWO1	PWO1password
PWO1	PWO1password
APS1	APS1password
APS2	APS2password
EWD1	EWD1password
EWD2	EWD2password
TPD(A)1	TPD(A)1

Table 3: LDAP Default Users

GROUP NAME	MEMBERS
APS	APS1, APS2
AWO	AWO1, AWO2
PWO	PWO1, PWO2
EWD	EWD1, EWD2
officers	CO1, AWO, PWO
Subsurfacedomain	CO1, PWO
Surfacedomain	CO1, PWO
Airdomain	CO1, PWO, AWO, APS, EWD, TPD(A)1

Table 4: LDAP Default Groups



5.3 User Configuration

5.3.1 LDIF File

The creation of new users and roles can be done through the importing of an LDIF file. This file defines users and groups, that users can be member of.

The first step is to set up users. Each user has a username and password as well as some required LDAP meta data. A set of example users is shown in Figure 2.

```
dn: cn=foo1, ou=isainUsers, o=dstl
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
cn: foo1
description: Foo person 1
sn: foo1
mail: foo1@dstl
userpassword: foo1password

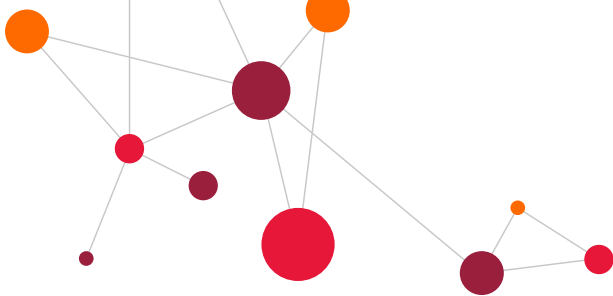
dn: cn=foo2, ou=isainUsers, o=dstl
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
cn: foo2
description: Foo person 2
sn: foo2
mail: foo2@dstl
userpassword: foo2password

dn: cn=bar1, ou=isainUsers, o=dstl
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
cn: bar1
description: Bar person 1
sn: bar1
mail: bar1@dstl
userpassword: bar1password
```

Figure 2: LDAP Users

The “dn” (distinguished name) field uniquely identifies each user, and contains their common name (cn), their organisational unit (ou) and their organisation (o). When creating new users and groups for ISAIN, ensure that the ou value is set as isainRoles and isainGroups respectively, and that the o value is set to dstl. Following this standard allows new users to work with the initial configuration of the rest of the applications in the network, allowing the users/groups to be accessed and used instantly. The objectclass fields define meta-data which ensure the entry is created as a person. It is essential that there is a single white line between each user, as this defines the end of one user and the beginning of another.

Once the users have been setup, they can be added to groups. Within ISAIN a user is given an ISAIN console role by adding them to a group. An example of a simple group is shown in Figure 3. The cn field is the name of the group and users are added to the group using the uniquemember field. It is important that the value of these fields match the dn field of each user exactly, otherwise the user will not be added. There is no limit to the number of uniquemember fields. Figure 4 shows a slightly more complex example of a group, where both a user and a whole group are added to another group. When adding a group to a group, all the members in the original group will then become members of the new group. This is a quick way of combining users together and can be useful when wanting to share a permission over several groups whilst keeping certain permissions separate.



```
dn: cn=foo, ou=isainGroups, o=dst1
objectclass: top
objectclass: groupofnames
cn: foo
description: a group of all foo users
uniquemember: cn=foo1, ou=isainUsers, o=dst1
uniquemember: cn=foo2, ou=isainUsers, o=dst1
```

Figure 3: LDAP simple group

```
dn: cn=everyone, ou=isainGroups, o=dst1
objectclass: top
objectclass: groupofnames
cn: everyone
description: a group with everyone
uniquemember: cn=foo, ou=isainUsers, o=dst1
uniquemember: cn=foo2, ou=isainUsers, o=dst1
uniquemember: cn=bar, ou=isainGroups, o=dst1
```

Figure 4: LDAP complex group

For ease when configuring the console application to use the LDAP server, ensure that the value given to 'ou' is the same for all users and groups (in this example, it is 'isainUsers' and 'isainGroups' respectively). It is also important to keep the value of 'o' the same for every object, be it user or group. It is also suggested that a 'main' or 'root' user is created, to act as the 'provider' user when configuring the console. This user shouldn't have any special permissions, and simply acts as an authorisation for the console application to connect to the LDAP server. A complete example LDIF file is shown in Appendix B – Example LDIF File.

Once the file has been created, it must be imported into the LDAP server. This is completed using either Apache Directory Studio or using 3rd party command line tools (outside the scope of this admin guide). From within Apache Directory Studio, select File > Import and choose the LDIF into LDAP option. A pop-up window will appear, as shown in Figure 5.

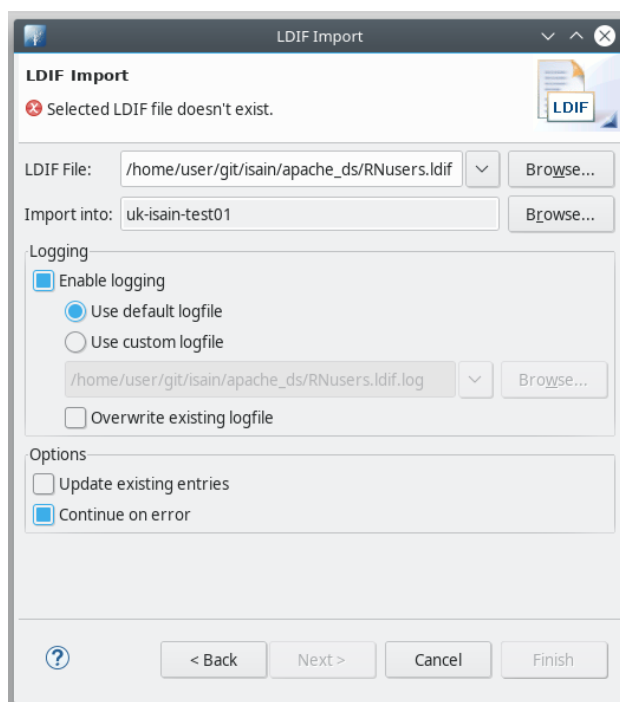
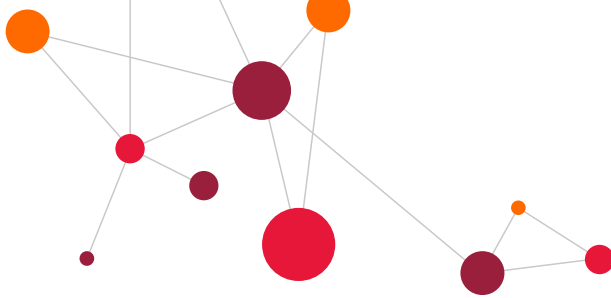


Figure 5: Apache Directory LDIF Import Window



Use the 'Browse' button to select the file, and make sure the 'Import into' option is set to the address of the LDAP server (if running on the same machine, leave as localhost). The 'Continue on error' option is best set as it will ensure that the data is imported, though it may be left off if there are uncertainties about the validity of the data. When all the options are set, click 'Finish' to import the users

5.3.2 Manual Configuration

Using an LDIF file is best suited for adding multiple users and groups to the LDAP server at once, however, if a single new user or group has to be added, it can also be done manually. This can also be done through Apache Directory Studio.

First, right click on "Root DSE" and select New > New Entry. This should bring up the window seen in Figure 6.

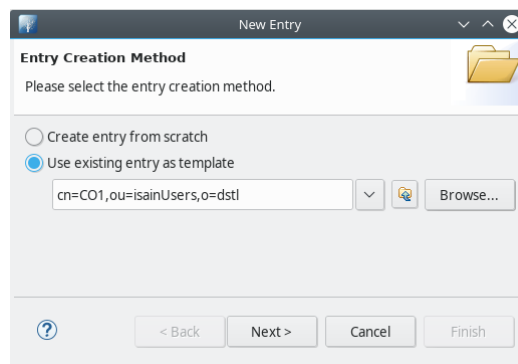


Figure 6: Apache Directory New Entry creation

Select the 'Use existing entry as template' option and select one of the already configured users, then click 'Next'. The next screen will display a list of attributes to associate with the new entry (object class, description, etc), including the attributes associated with the entry used as template. Use the 'Add' and 'Remove' buttons to select the relevant attributes for the desired entries, using the examples above to select which attributes are required, as shown in Figure 7.

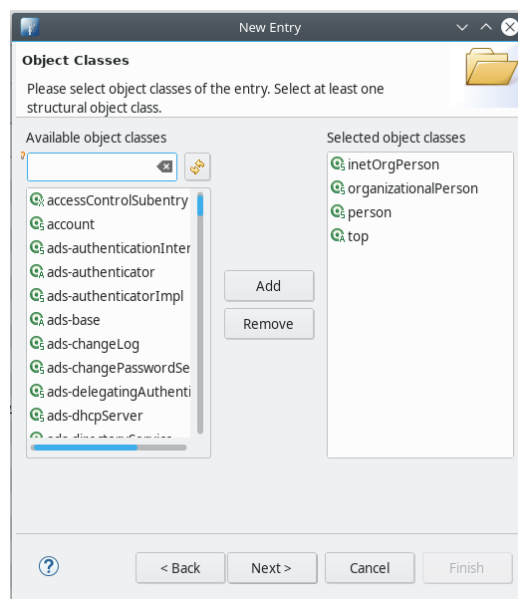
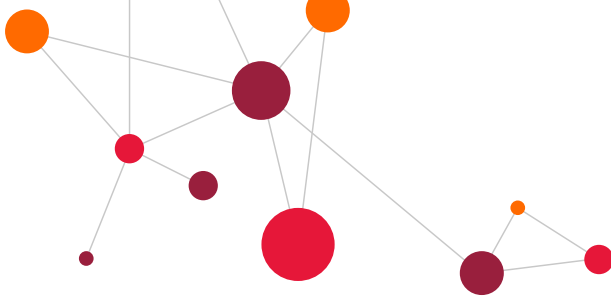


Figure 7: Apache Directory attribute selection



Once all the attributes have been selected, click 'Next'. In the next window, change the cn value for the new entry, ensuring it is not a duplicate of any existing entries, ensuring the DN Preview updates correctly, as shown in Figure 8.

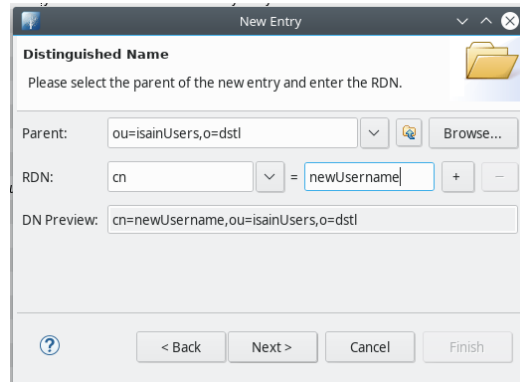


Figure 8: Apache Directory DN Configuration

Once the value has been set correctly, and the dn value matches the expected dn for the type of entry being created, click 'Next'. The final screen involves setting values for each of the attributes selected earlier. Use the examples previously to deduce what the value for each attribute should be. Once complete, the window should look like Figure 9.

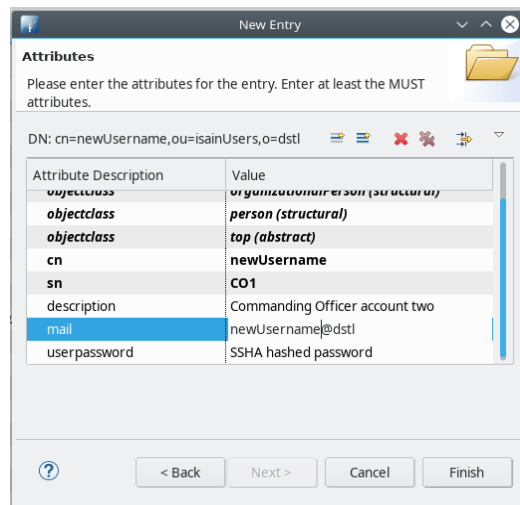


Figure 9: Apache Directory Attribute configuration

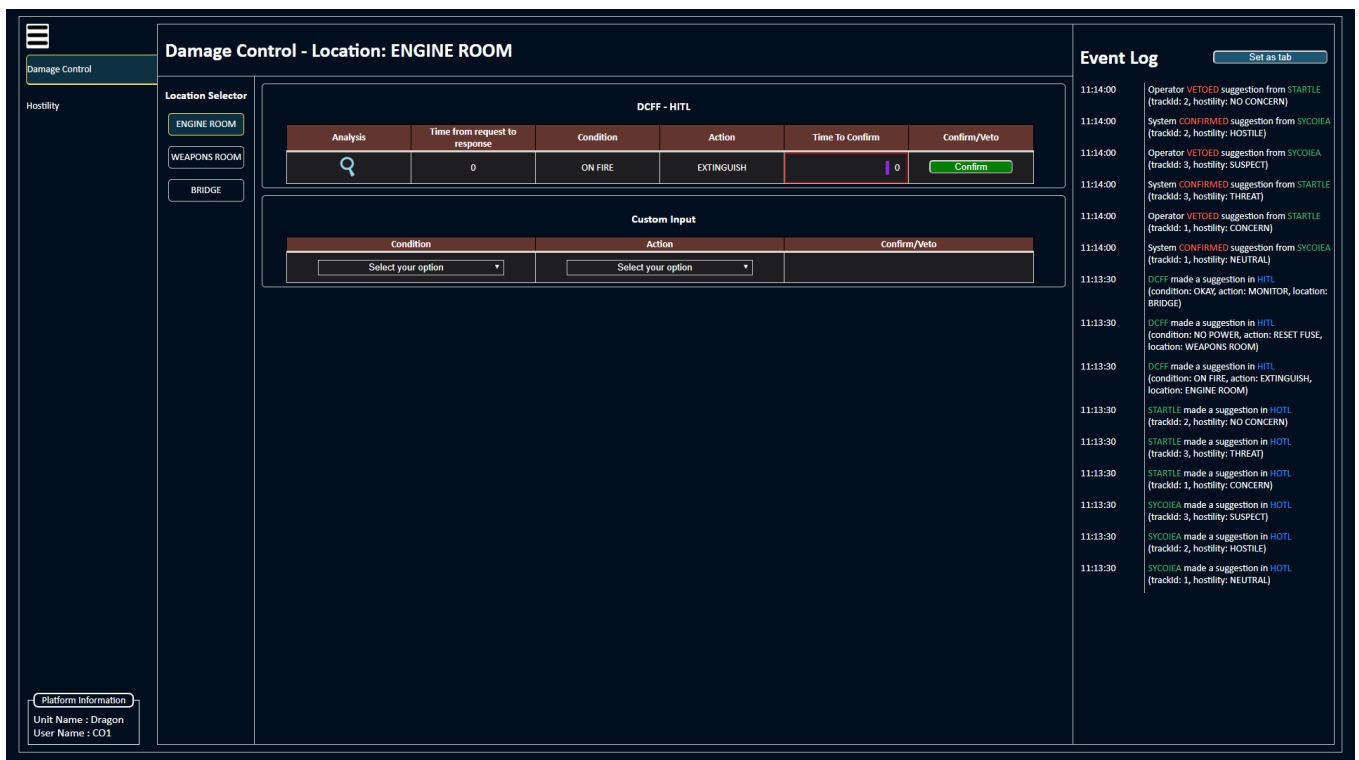
Finally click 'Finish' to complete the process. The new entry will be added to the server, and the credentials can now be used to enter the console. Ensure that any groups the user should be a part of are also updated following this same process, but with the Edit Entry wizard. See the online Apache Directory Studio guide for more information.

To add this new user to an existing group, navigate to the group entry within Apache Directory, and select the Edit Entry option from the menu. The steps to add the user are the same as some of the previous steps (Figure 7 and Figure 9). Adding and setting a value for the new attributes is done in the same manner as before, except for adding a new member, just add a new uniquemember attribute for each new member to add, and set the value for these attributes to be the dn value for the newly created user. Once this process is complete, the user will be part of the group, and have the permissions and accesses that the group provides within the console application.

6 Console

6.1 Overview

The console serves as a window into the ISAIN processes, allowing the operator to view the responses made by each connected AI, as well as providing their own input by deciding which AI response should be fed back into the network. To facilitate this process, the operator can view the justifications an AI used to arrive at its decision as well as comparing decisions and justifications from different AIs which have been given the same task.



The screenshot shows the console interface for 'Damage Control - Location: ENGINE ROOM'. On the left, there is a 'Location Selector' with buttons for 'ENGINE ROOM', 'WEAPONS ROOM', and 'BRIDGE'. Below it is 'Platform Information' showing 'Unit Name : Dragon' and 'User Name : CO1'. The main area is divided into two sections: 'DCFF - HITL' and 'Custom Input'. The 'DCFF - HITL' section contains a table with columns for Analysis, Time from request to response, Condition, Action, Time To Confirm, and Confirm/Veto. The 'Custom Input' section contains a table with columns for Condition, Action, and Confirm/Veto. On the right, there is an 'Event Log' with a 'Set as tab' button and a list of events with timestamps and descriptions.

Analysis	Time from request to response	Condition	Action	Time To Confirm	Confirm/Veto
Q	0	ON FIRE	EXTINGUISH	0	Confirm

Condition	Action	Confirm/Veto
Select your option	Select your option	

Event Log

- 11:14:00 Operator VETOED suggestion from STARTLE (trackid: 2, hostility: NO CONCERN)
- 11:14:00 System CONFIRMED suggestion from SYCOEA (trackid: 2, hostility: HOSTILE)
- 11:14:00 Operator VETOED suggestion from SYCOEA (trackid: 3, hostility: SUSPECT)
- 11:14:00 System CONFIRMED suggestion from STARTLE (trackid: 3, hostility: THREAT)
- 11:14:00 Operator VETOED suggestion from STARTLE (trackid: 1, hostility: CONCERN)
- 11:14:00 System CONFIRMED suggestion from SYCOEA (trackid: 1, hostility: NEUTRAL)
- 11:13:30 DCFF made a suggestion in HITL (condition: OKAY, action: MONITOR, location: BRIDGE)
- 11:13:30 DCFF made a suggestion in HITL (condition: NO POWER, action: RESET FUSE, location: WEAPONS ROOM)
- 11:13:30 DCFF made a suggestion in HITL (condition: ON FIRE, action: EXTINGUISH, location: ENGINE ROOM)
- 11:13:30 STARTLE made a suggestion in HOTEL (trackid: 2, hostility: NO CONCERN)
- 11:13:30 STARTLE made a suggestion in HOTEL (trackid: 3, hostility: THREAT)
- 11:13:30 STARTLE made a suggestion in HOTEL (trackid: 1, hostility: CONCERN)
- 11:13:30 SYCOEA made a suggestion in HOTEL (trackid: 3, hostility: SUSPECT)
- 11:13:30 SYCOEA made a suggestion in HOTEL (trackid: 2, hostility: HOSTILE)
- 11:13:30 SYCOEA made a suggestion in HOTEL (trackid: 1, hostility: NEUTRAL)

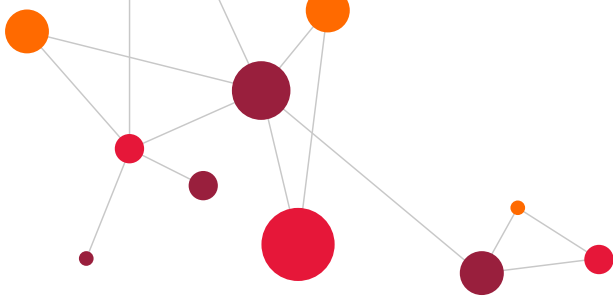
Figure 10: Console overview

The console is accessed through a web browser using the URL `http://<server name>:8085` and will run on any modern web browser, though Google Chrome is preferred. If the browser supports it, the console is best viewed using the full screen, as this will ensure the information displayed is clear and understandable to the operator

6.2 Initial Configuration

To ensure the initial configuration of the console is as simple and straight forward as possible, the configuration is mostly kept in one file, named `application.properties`, found in `/opt/isain/console/config`. Here, configuration exists for accessing both the LDAP server and MongoDB, as well as setting the addresses for components within the console and values that are displayed and used on the User Interface (UI) of the console. A majority of the configuration is set to work with the default components of the network, and only need modifying if substituting in a new components. However, this section will still cover all this configuration, as well as the other necessary configuration.

If using the ApacheDS LDAP server and users/groups provided with the ISAIN setup, as explained in section 5, no changes need to be made to the LDAP configuration. However, if another LDAP server is substituted in, the



corresponding console configuration options must be updated to reflect the changes. There are five properties that must be configured for the console to properly access the users and groups. They are as follows:

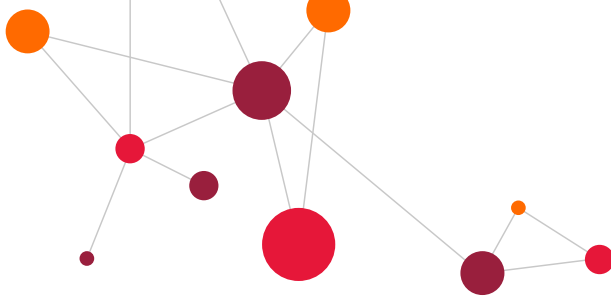
- `ldap.provider.url` – this is the URL of the LDAP server, of the form `ldap://{IP-ADDRESS}:{PORT}`. Ensure that the URL is prepended with 'ldap', not 'http'.
- `ldap.provider.userdn` – this is the user that is used to authenticate and access the server, known as the 'provider'. The value provided should match the 'dn' of the 'main' or 'root' user created within the LDAP configuration.
- `ldap.provider.password` – this is the password of the user being used as the 'provider'.
- `ldap.user.dn.patterns` – this defines the query used when finding all the users available. The 'cn' value should be left as '{0}', and the 'ou' and 'o' values changed to reflect the values set in the configured users.
- `ldap.group.search.base` – this defines the query used when finding all the groups a user is a part of. The 'ou' value should match the 'ou' value chosen to represent groups, and the 'o' value should match in the same manner.

Ensure that the `security.server.address` and `ui.server.address` are set to the same value, and the IP matches the IP of the machine running the console docker container. These properties should be of the form `http://XXX.XXX.XXX.XXX`, where `XXX.XXX.XXX.XXX` is the machine's IP address. Changing the ports requires changing the value in two separate locations. For the UI server, set the `ui.server.port` value in `application.properties`, and `server.port` in `ui.properties`, ensuring they are equal. For the security server, set the `security.server.port` value in `application.properties` and the `server.port` value in `security.properties`, once again ensuring they are equal.

The `spring.data.mongodb.uri` value defines the address of the MongoDB database. If using the default MongoDB container, then this value does not need changing. However, if a different MongoDB instance is used, this value will need to be updated. When doing so, ensure that the address is prepended with `mongodb://`.

The last set of properties in the file are for the UI. The first, `ui.refresh.rate` determines the period at which the UI updates all its values, in seconds. The lower this value, the more often the UI refreshes. However, a lower value can lead to performance issues, so experimenting with different values is suggested. The default value of 1 second is often sufficient for this purpose.

The `nifi.url` and `kibana.url` should be set to the IP address of the machine that the ISAIN network is running on so that the links within the console reference the correct addresses. When making this change, only change the IP address – do not change the port number, as this will remain the same.



6.3 Configuring tabs

Each tab on the console displays the data associated with a given domain, be it hostility, damage control, etc. Whenever an AI is added to the network that will be giving responses in a new domain, a new tab must be configured to handle and display this domain. Two default tabs are included, and any new tabs must be added to the 'tabs' collection of the 'isain' database in the MongoDB database. This can be done either through MongoDB Compass Community or via the command line with an update script.

Tabs are defined in JSON, a commonly used data format that allows complex structures to be built in an easy to understand fashion. Figure 11 shows an example tab configuration, showing the fields that need to be set to correctly configure the tab.

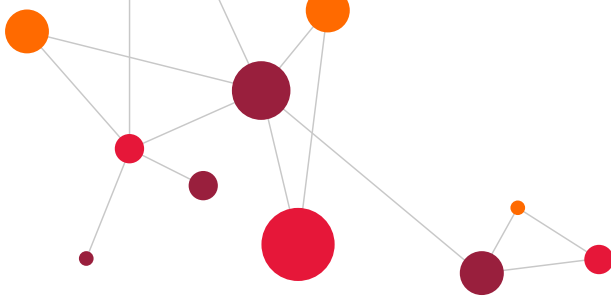
```
{
  "id": "DAMAGE_CONTROL",
  "title": "Damage Control",
  "domain": "damageControl",
  "exclusive": true,
  "headers": [{
    "displayName": "Condition",
    "key": "condition",
    "selectable": true,
    "aiOptions": {
      "IBIS": [
        "RUNNING",
        "FAULTY",
        "NO POWER"
      ]
    }
  }], {
    "displayName": "Action",
    "key": "action",
    "selectable": true,
    "aiOptions": {
      "IBIS": [
        "MONITOR",
        "INVESTIGATE",
        "RESET FUSE",
        "EXTINGUISH"
      ]
    }
  }
}, {
  "selectorKey": "location",
  "authorities": []
}
}
```

Figure 11: Example Tab configuration

The tabs are stored in the 'tabs' collection of the database, and adding a new tab is as simple as adding a new entry to this collection.

It is best to create a new tab by copying one of the existing default tabs. This way, all the required fields are available and just need to be modified accordingly. The required fields and their purpose are:

- "id" – a string that identifies the tab. Must be unique, with no whitespace, but can take any form otherwise. Convention is that the "id" is equal to the "title", but capitalised and substituting any white spaces for an underscore character "_".
- "title" – the title of the tab, as appears on the console.
- "domain" – the domain of the tab. This must match the domain specified in the NiFi flow configuration to ensure the correct data is displayed on the console.



- “exclusive” – should be set to either true or false. If true, then confirming one response from an AI for this domain will automatically veto any other responses in the same group. If false, the operator can confirm as many responses in a group as they wish.
- “headers” – this contains an array of header objects, which describe how the data specific to the chosen domain should be interpreted and displayed on the console. The attributes associated with a header are as follows:
 - “displayName” – the name that the column will have for this header.
 - “key” – the key of the associated data found in the data specifics for this domain. Must match the key set in the NiFi configuration.
 - “selectable” – this attribute is optional, and if not included, will default to false. If false, then the operator will not be able to add a custom value for this data, whereas if true, the operator will have the option to pick a custom option.
 - “aiOptions” – this attribute is only required if “selectable” is set to true. This attribute is a list of possible options for this data, grouped by the AI the options belong to. An example configuration for this option is shown in Figure 12.

```

"headers": [{
  "displayName": "Hostility",
  "key": "hostility",
  "selectable": true,
  "selectorType": "OPTIONS",
  "isainOptions": {
    "SYCOIEA": [
      "FRIENDLY",
      "NUETRAL",
      "SUSPECT",
      "HOSTILE"
    ]
  }
}]

```

Figure 12: Example headers configuration

- “selectorKey” – the selectorKey refers to an attribute specific to the domain that the responses will be grouped by. The value must match the attribute, including the same case, for the grouping to work correctly
- “sortSelector” – if set to true, the selector for this tab on the console will be sorted numerically, in ascending order.
- “authorities” – an array of the groups that an operator must be a part of to access this tab, as defined in the LDAP configuration. If left empty, any operator will be able to access the tab. Each role should be wrapped in quotes “” and will be of the form ROLE_{LDAP-GROUP-NAME}, all in upper-case. Figure 13 shows an example of the format, if the groups configured in the LDAP server are ‘officers’ and ‘airdomain’.

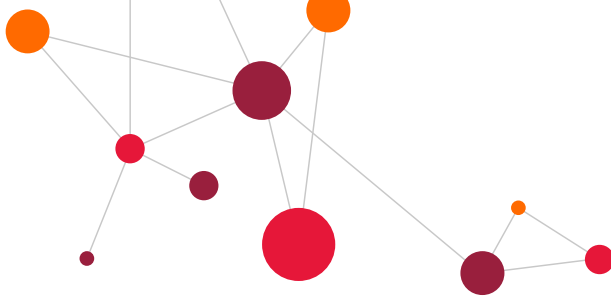
```

"authorities": [
  "ROLE_OFFICERS",
  "ROLE_AIRDOMAIN"
]

```

Figure 13: Example authorities configuration

Once creating a new tab, ensure the file is saved correctly. If the console is running, it will need to be restarted and then the browser refreshed before the changes will take effect. If configured correctly, the new tab should appear and when data is sent to the domain, it should be displayed as expected. If there are issues with data not



being displayed, then check the tabs configured and ensure they are valid. Most software for accessing MongoDB databases contain their own validation, and will alert you if something is incorrect. Ensure that all the mentioned fields are present and match the default tabs provided.

To edit the tab configuration within MongoDB Compass and add a new tab, first connect to the database as shown in Figure 14.

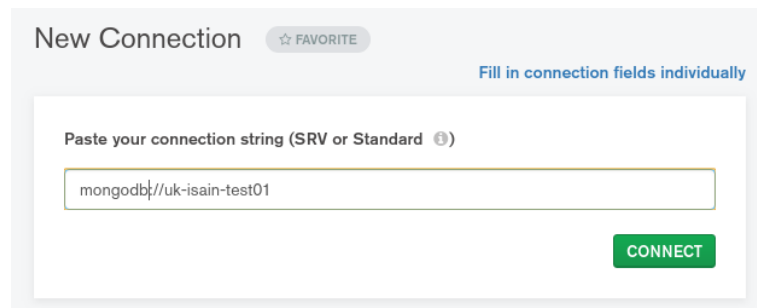


Figure 14: MongoDB Compass connection

Next select the “tabs” collection and select “ADD DATA -> Insert document” as shown in Figure 15.

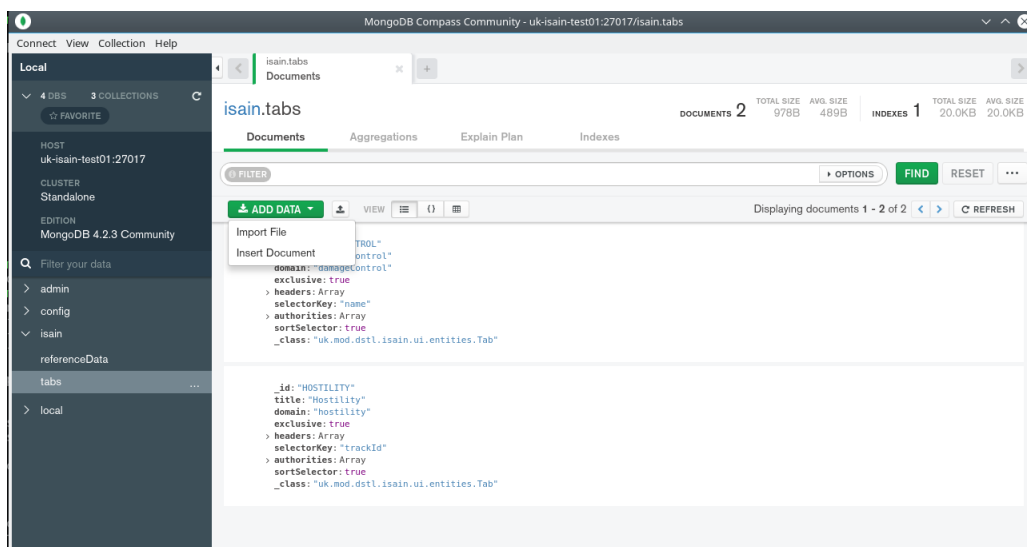


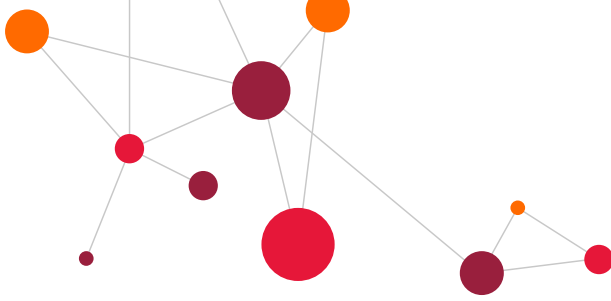
Figure 15: MongoDB Compass add document

Alternatively, the new tab can be added to an update script that can be executed on the server running the MongoDB Docker container as shown in Figure 16.

```
docker exec -i isain_mongodb_1 mongoimport -c=tabs -d=isain < <tab config file>.json
```

Figure 16: Running the mongoimport command line tool

The mongoimport command has many other options, such as allowing the import of a JSON array in a file, and these options can be found with the online mongoimport documentation.



7 Apache NiFi

7.1 Overview

Apache NiFi provides the data transformation and routing that forms the backbone of ISAIN. A NiFi flow routes input data to the relevant AI(s), passes responses to the console and then feeds the operator's decision back into the network. Data packets within NiFi are referred to as FlowFiles, and consist of any number of attributes and the content block, which can take any number of formats, including JSON, XML, etc. All aspects of these FlowFiles can be modified and accessed by NiFi Processors, allowing complex transformations of data within the flow.

The provided NiFi flow handles most of the tasks required by default, but some extra configuration is needed to ensure that certain functions are handled by the flow, especially when adding new AIs. This section will cover the steps that must be taken to ensure the application works correctly and performs the expected tasks.

The NiFi configuration page can be accessed from either the ISAIN console or directly by entering the NiFi Uniform Resource Location (URL), which by default is `https://<ISAIN HOSTNAME>:18085/nifi`. Before continuing, ensure you are familiar with how to navigate and edit the NiFi flow. For information on how to do this, please refer to the Apache NiFi User Guide referenced in Table 2: Software stack. A key part of the NiFi User Guide is the section on Data Provenance, as this will explain how to examine the movement of FlowFiles through any given processor, which is a useful tool when debugging issues within the network, as being able to view the contents of the file, and replay its movement through the network can help explain the issue being investigated.

7.2 Policy Management

The NiFi application included with ISAIN is secured using the same LDAP server used for the Console. This allows control of both access to the application as a whole, as well as providing policies determining how much users can do within the application.

When accessing the application, the user will be prompted to login, using the same username and password they used for the Console. Once within the application, policies are separated into two groups: Policies and Access Policies.

Policies, accessed from the Menu in the top-right of the NiFi interface, define global permissions for users. These include viewing and modifying the interface, modifying the policies and viewing data in the flow. Access Policies are defined on a Processor/Processor Group, and control what users can access that particular object. Access Policies can be configured by right-clicking on the Processor and selecting the 'Manage Access Policies' option. These policies define how users can interact with that particular processor, and must be set along with the global Policies to allow a user to properly view, modify and query processors and data. It is important to remember that Access Policies applied to a Processor Group will be shared with all the components existing in that group.

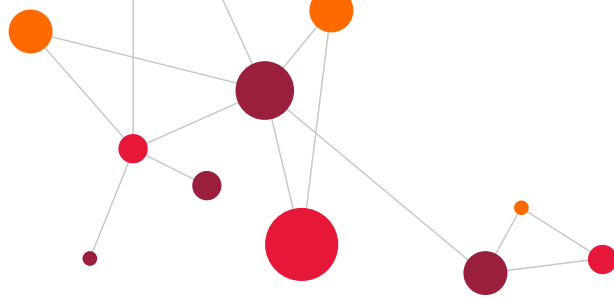
For more information on policies, their management and properties and configuring NiFi to use an LDAP server, consult the NiFi Administration Guide, which can be found online.

7.3 Variable Configuration

NiFi allows variables to be used to share a particular value across the flow that can be configured from a single place. The NiFi flow provided has three variables set, defining important values for interacting with the MongoDB database. If the default MongoDB application provided is used, then these values will not need updating.

However, if a different MongoDB instance is used, these variables will need updating accordingly. They are:

- `mongo_uri`: this is the address of the MongoDB application. Ensure that when updating, the address is prepended with 'mongodb://'. By default this is 'mongodb://isain-svr:27017'



- `mongo_database`: this is the database within MongoDB being used. By default this is 'isain'
- `mongo_output_collection`: this is the collection that actioned responses are passed to. From here, they can be fed back into connected AIs. By default this is 'effectors'

Once these variables are set, the flow will begin using the instance of MongoDB they reference. Whenever adding new processors, etc, ensure to reference these variables to make changing the values far easier when needed.

7.4 Networked (HOOTL) and Non-Networked (HITL) Configuration

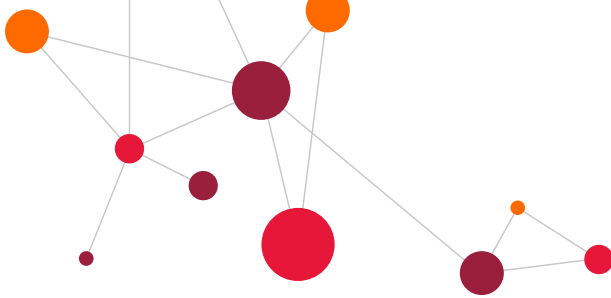
The ISAIN console operates as a HITL buffer between the automatic interactions between AIs. This is to demonstrate 'non-networked' mode, in order to highlight what decision processes ISAIN can remove from the user. In order to process these interactions, everything is contained within the 'Console Flows' processor group in the ISAIN NiFi flow. ISAIN is by default in 'networked' mode and so any communication between AIs is entirely human out-of-the-loop (HOOTL). As AIs interact automatically, all communication is done entirely within the NiFi flow. The console simply displays the 'confirmation' of the interactions between the AIs, with no ability for the user to veto/confirm themselves.

The management of what is sent to the console from NiFi, and vice versa, the HITL processor group continuously checks the MongoDB database for any responses that have expired, and automatically confirms/vetoes the relevant responses according to their grouping and priority.

7.5 Simulation Environment

Virtual Battle Space 3 is currently used as the simulation environment of the real-world, containing the ISAIN ship, and surrounding vessels. Simulation Gateway obtains the relevant data from the simulation in order to generate an input of tracks into the network. Each of these tracks has a huge array of attributes that describe every feature about it, allowing, for example, a connected AI to make an informed decision about its hostility.

Simulation Gateway outputs the system tracks as DDS messages (see Section 10.2.2.1), to be consumed by the NiFi dataflow and distributed to any connected AIs.



8 Elastic Stack

The Elastic Stack, formerly known as the ELK stack, is a combination of three pieces of software that together provide a powerful but easy to use way to collate and analyse log files from a huge amount of different applications. The stack consists of:

- **ElasticSearch:** Stores the logs sent to the stack, and provides efficient and effective indexing, searching and analysis tools.
- **Kibana:** Kibana is a powerful data visualisation tool that makes the most of the tools provided by ElasticSearch to provide data visualisations for the user to greater understand the data stored. Kibana can also be used to display the raw logs in an easy to query manner.
- **LogStash:** Serves as a pipeline to send logs from various input sources to ElasticSearch in an efficient manner, allowing the balancing of a huge amount of different source.

Along with these components, FileBeat is also used to read in the individual log files of each component of the network and pass them onto LogStash, which in turn passes them to ElasticSearch.

8.1 Adding a new component

The ElasticStack provided with ISAIN is configured to read and index all the logs from each component. However, if a new component is needed to be added, then some small amount of configuration is required to pass the logs into the ElasticStack, with the majority of the changes being made in FileBeat.

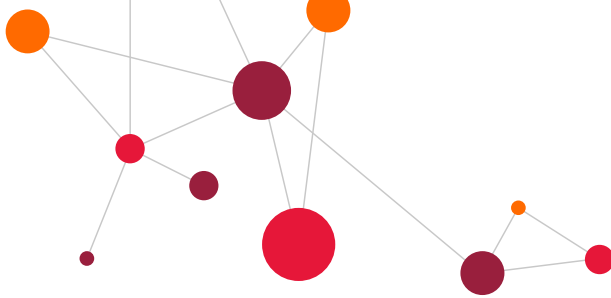
The first step is to map the log files into the FileBeat Docker container. This can be done by editing the docker-compose.xml file used to start/stop the network, found in /opt/isain. An example configuration of the FileBeat container in this file can be seen in Figure 17.

```
filebeat:
  image: 'uk-isain-proj01.rcnet.groupinfra.com:8082/store/elastic/filebeat:7.6.0'
  command: /usr/local/bin/docker-entrypoint --path.logs=/var/log/isain/filebeat
  volumes:
    - /opt/isain/elk/filebeat/filebeat.yml:/usr/share/filebeat/filebeat.yml:z
    - /opt/isain/nifi/logs:/var/log/isain/nifi:z
    - /opt/isain/mongodb/logs:/var/log/isain/mongodb:z
    - /opt/isain/elk/kibana/logs:/var/log/isain/kibana:z
    - /opt/isain/apacheds/logs:/var/log/isain/apacheds:z
    - /opt/isain/elk/logstash/logs:/var/log/isain/logstash:z
    - /opt/isain/zookeeper/logs:/var/log/isain/zookeeper:z
    - /opt/isain/elk/elasticsearch/logs:/var/log/isain/elasticsearch:z
    - /opt/isain/console/logs:/var/log/isain/console:z
  environment:
    - path.logs=/var/log/isain/filebeat
  user: root:root
  network_mode: host
  extra_hosts:
    - "isain-svr:0.0.0.0"
```

Figure 17: FileBeat Docker Configuration

To map in a new log file / directory, a new entry will need to be added to the 'volumes' entry in the configuration. Following the format of the other entries, including the amount of whitespace. Entries in the table are of the format <Path to logs/directory on host>:<Path for logs/directory in container>:z. Consult the Docker Compose documentation for more information. Make sure to remember the path the logs are found in the container, as this is needed in the next stage of the configuration.

Once the logs have been mapped into the container, FileBeat needs to be configured to detect and send the logs to LogStash. This configuration is done within the filebeat.yml file, found at /opt/isain/elk/filebeat/filebeat.yml.



Within this file, there is a section called 'filebeat.inputs', where logs to read in are defined. An example configuration can be seen in Figure 18.

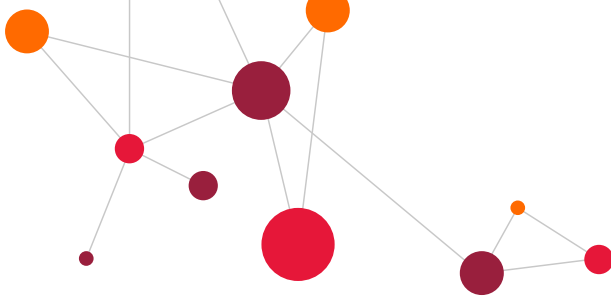
```
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - /var/log/isain/nifi/nifi-app.log
    - /var/log/isain/nifi/nifi-user.log
    - /var/log/isain/nifi/nifi-bootstrap.log
  fields:
    application: nifi
    fields_under_root: true
    scan_frequency: 5s
    close_inactive: 10m

- type: log
  enabled: true
  paths:
    - /var/log/isain/mongodb/*.log
  fields:
    application: mongo
    fields_under_root: true
    scan_frequency: 5s
    close_inactive: 10m
```

Figure 18: FileBeat configuration example

Each entry in the filebeat.inputs list defines where a selection of log files can be found, and sets some important information before forwarding to LogStash. When adding a new component, it is suggested that a pre-existing entry be copy and pasted, as this keeps the format correct and saves time, as only a few changes need to be made. The first is adding entries to the 'paths' field for each log file/directory that was mapped in previously. Make sure the path matches, and use the '*' operator to match all the files in a directory. The second is the 'application' field in the 'fields' section. This is a custom added field, and should be set to the name of the application/component. This makes filtering the logs within Kibana easier.

Once the changes have been made, the FileBeat container will need restarting for the changes to be detected and the logs forwarded.



9 ISAIN Dynamic Dataflow Configuration (IDDC)

The purpose of the ISAIN Dynamic Dataflow Configuration (IDDC) is to better integrate a new AI into ISAIN, via the NiFi dataflow, which does not require identifying and implementing specific interactions with pre-existing AIs. It allows ISAIN to automatically determine which AI to invoke from an arbitrary input that conforms to the Planning Domain Definition Language (PDDL) structure. Preventing the requirement to design strict hard-wired connections between AIs, allowing for more complex dataflows to be generated automatically.

The IDDC utilises the Fast Downward “domain-independent classical planning system” in order to find the most appropriate path to route data through ISAIN by taking a ‘domain’ and ‘problem’ description each in the form of a PDDL configuration. The ‘domain’ represents the set of rules to adhere to, and the ‘problem’ representing the start and end state.

The IDDC within NiFi is able to consume from a source and route the required data to AIs already within ISAIN. Currently the IDDC is set to route system tracks taken from Simulation Gateway, which are a representation of the object within the simulated environment. Significantly more functionality has been incorporated into the dataflow inferer, but has not been implemented for this phase of the project.

As an example for routing system tracks, given the below domain and problem files in Figure 19 and Figure 20, the IDDC is able to consume the Simulation Gateway system track and route it to the required AIs. The IDDC has a universal input port, and given more domain and problem files the IDDC will be able to route more complex dataflows through ISAIN automatically.

```
System Track Domain PDDL
{
  "dataflowname": "systemtrack",
  "domain":
  "(define (domain simpleRouting)
  (:predicates (dataItem ?di) (source ?s) (destination ?d) (dataItemAt ?di ?d))
  (:action routeDataItemToApplication
  :parameters (?di ?s ?d)
  :precondition (and (dataItem ?di)
                    (source ?s)
                    (destination ?d))
  :effect (and (dataItemAt ?di ?d)))"
}
```

Figure 19: Example Domain PDDL entry

```
System Track Problem PDDL
{
  "dataflowname": "systemtrack",
  "problem": "(define (problem systemTracksSimple)
  (:domain simpleRouting)
  (:objects simulationGateway sycoiea knot tacnav systemTrack)
  (:init (dataItem systemTrack)
         (source simulationGateway)
         (destination sycoiea)
         (destination knot)
         (destination tacnav)
  )
  (:goal (and (dataItemAt systemTrack sycoiea)
             (dataItemAt systemTrack knot)
             (dataItemAt systemTrack tacnav)))
  )"
}
```

Figure 20: Example Problem PDDL entry

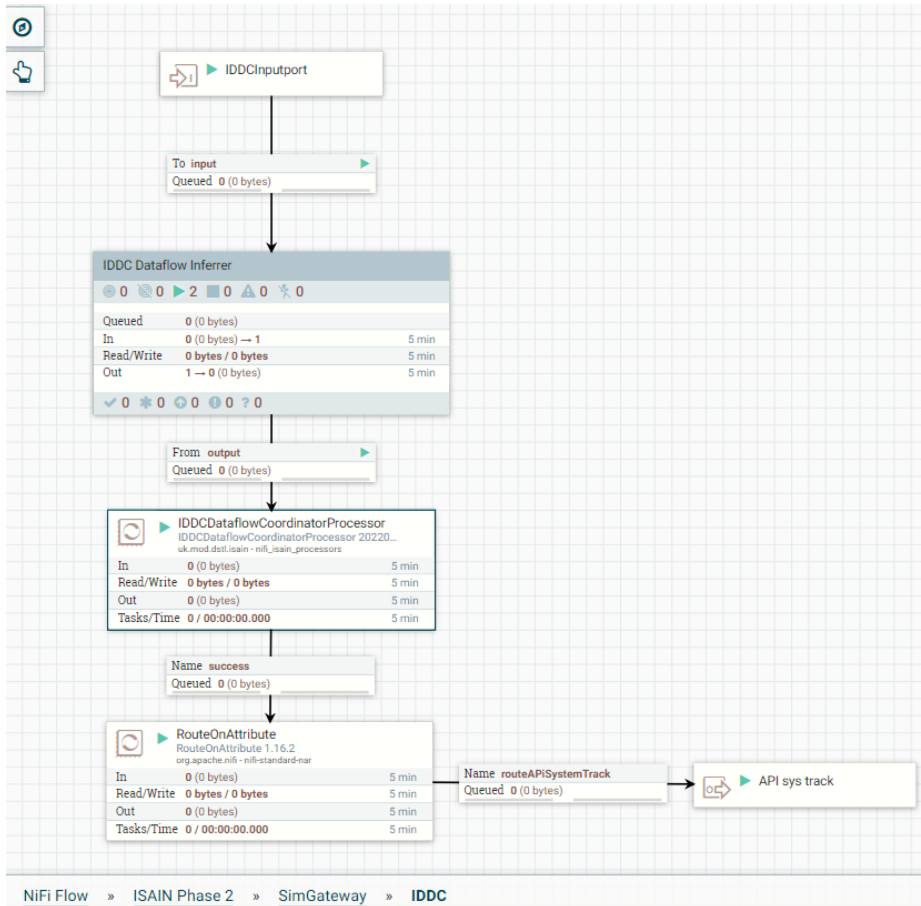
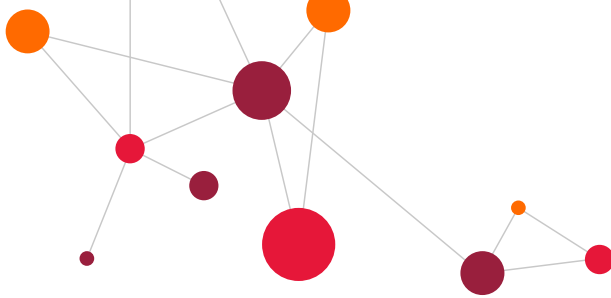
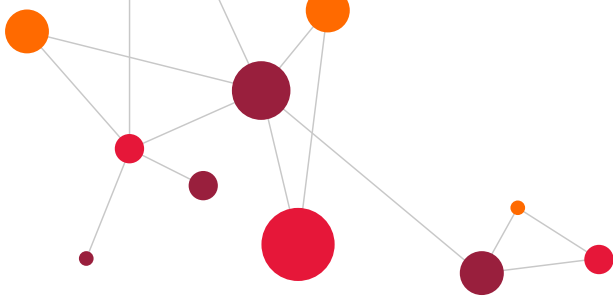


Figure 21: Example IDDC NiFi flow



10 AI Integration

10.1 Overview

A key feature of ISAIN is the ability to integrate other applications and AIs into the network, including any future projects. AIs can be added to the network at any point and with minimal configuration, once the integration work is complete. This section will cover the steps taken to add both AIs and Synthetic Environments to the network.

10.2 AI Integration

When integrating new AIs into the network, there are two features that must be considered:

- AI input/output format
- AI communication methods

It is important that each of these points are met, as only then will the AI interact correctly with the network. Each point can be achieved via:

1. Existing NiFi processors and ISAIN NiFi extensions; or
2. The addition of new NiFi extensions in combination with (1); or
3. Modification of the AI source code in combination with (1).

This guide only covers using existing NiFi processors and ISAIN NiFi extensions. If the addition of new NiFi extensions is required, refer to the ISAIN Developer Guide [\[1\]](#).

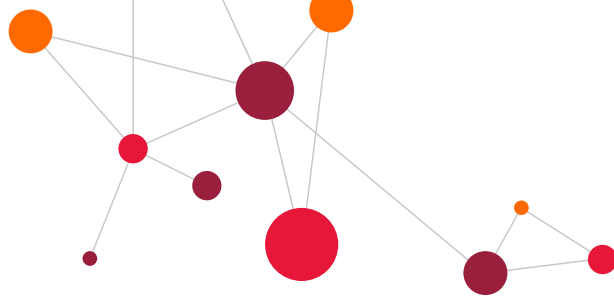
When integrating, it is good practise to make use of the Processor Groups within NiFi, which allow the creation of a 'black box', where any communication and transformation for the AI are handled away from the main NiFi flow, with just input and output ports being exposed to access the AI. This method allows AIs, once integrated, to be connected and disconnected from the network in a consistent manner, meaning system administrators do not require expert knowledge of how the AI is integrated to understand how to plug the AI into the network.

10.2.1 AI input/output format

The network, where possible, attempts to use common data formats found within the domain of ISAIN, most commonly the OACS datatypes, such as SystemTrack, etc. Using these standardised formats is beneficial, as it allows communication between the network and connected applications to be simple and consistent. Most of the AIs within this domain that will be integrated with ISAIN are most likely already using these datatypes, but if this is not the case, the input data must be transformed before passing to the AI, and the AI's output must also be transformed into the network specific format. Within the network, ISAIN expects data to be in XStream XML, but in the JSON format.

NiFi can handle this data transformation using either the default NiFi processors the ISAIN DataConversionProcessor. NiFi includes processors that can be utilised to transform data from one format to another, such as:

- EvaluateJSONPath – if the content of the FlowFile is in JSON, can be used to move JSON attributes to FlowFile attributes.
- JoltTransformJSON – uses the Jolt specification to transform in place the FlowFile content, when in JSON. The Jolt specification allows adding and removing attributes, as well as shifting attributes and other transformations, and all these transformations can be done at once, making this processor powerful when dealing with JSON data.



- UpdateAttribute – NiFi includes the NiFi Expression Language, which is a basic set of tools that can perform numerical, string and boolean transformations on FlowFile attributes, both in place and setting new attributes. Is useful with EvaluateJSONPath and JoltTransformJSON when a more complex transformation is required on a JSON attribute.
- ReplaceText – Finds and replaces text within the FlowFile content using a Regular Expression. This allows either the whole content to be replaced, or only specific parts.

The online resources for these processors are the best source of information on how to utilise them correctly. It is also worth using the online resources to familiarise with the NiFi Expression Language, as this makes these processors even more powerful.

The DataConversionProcessor uses convertor classes, implementing a common parent interface, to convert data from one format to another. This processor can be used at any point, and transforms the input data, in JSON, to another format, which is also outputted as JSON. This processor is ideal for making complex transformations in a single operation instead of chaining together multiple processors. As mentioned previously, the DataConversionProcessor expects input in XStream XML as JSON. For information on how to create new convertors and import them into NiFi, refer to the ISAIN Developer Guide [\[1\]](#).

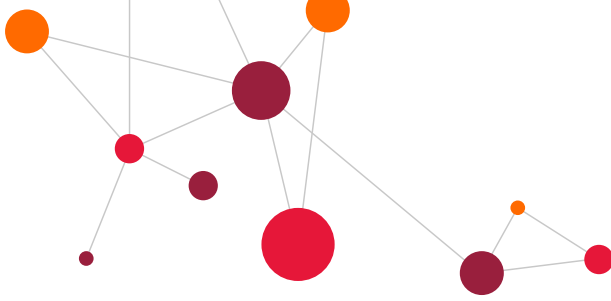
When integrating with the NiFi flow, all responses must be sent to the AI Responses input port of the Console Flows processor group provided with ISAIN. Before data is routed here, it is essential that a number of attributes are set on the FlowFile beforehand. These attributes can be set using a combination of EvaluateJSONPath and UpdateAttribute processors. The attributes are as follows:

- ai: the name of the agent who made the response.
- analysisData: any justification data provided by the agent to explain its decision – must be in JSON format.
- inputData: the input data provided for the responses – must be in JSON format.
- dataSpecifics: this is the response itself, including the 'selector' field and value, as well as any other important information associated with the data. Must be in JSON format.
- domain: the domain the response is in, should match the domain value of the corresponding tab.
- teaming: the automation the agent is in, either HITL or HOOTL.
- timeRequested: a timestamp (milliseconds since epoch) of when the request was made to the agent.
- timeReceived: a timestamp (milliseconds since epoch) of when the response was received.
- ttc: Time To Confirm. A timestamp (milliseconds since epoch) of when the response should be actioned. Due to the bug referenced in **USER GUIDE** if running in HITL, the ttc is set to 15 minutes so the user will have time to action a decision before an automatic conformation

10.2.2 AI communication methods

NiFi, by default, includes a number of input and output processors capable of communicating across a range of common protocols including, amongst other, REST, AMQP and JMS. If for example the AI being integrated uses a RESTful interface, the InvokeHTTP processor can be used to called the AI's endpoints and pass the response back to the network

Where an AI is to be integrated into ISAIN and none of the supplied NiFi input and output processors are suitable, custom processors can be added to NiFi. Refer to the ISAIN Developer Guide [\[1\]](#) for details.



10.2.2.1 DDS

In addition to the standard NiFi processors, ConsumeDDS and Publish DDS input and output processors have been added to subscribe and publish to DDS topics, using the OpenSplice DDS implementation. As DDS is a cross vendor industry standard, these processors should work regardless of the DDS vendor used by the AI. The data type and QOS profile names are specified as properties on the processors allowing for multiple DDS connections across different DDS domains and topic. An example of the DDS properties is show in Figure 22.

Property	Value
Domain ID	0
Topic	No value set
Partition names	No value set
Data type	No value set
Domain_Participant_Qos	No value set
Topic_Qos	No value set
Qos_Library_Name	No value set

Figure 22: NiFi DDS processor properties

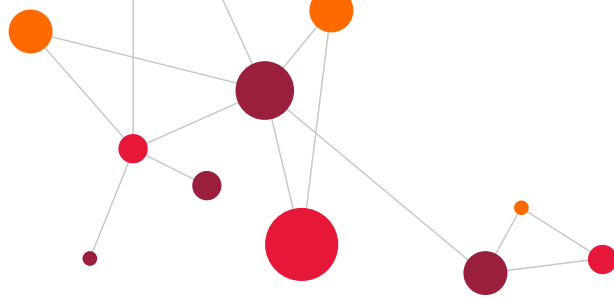
The QOS profiles are defined within the XML file /opt/nifi/nifi-current/ USER_QOS_PROFILES.xml within the NiFi Docker container. This file can be updated either by replacing it with a new version using the command shown in Figure 23, or by overriding it with a file on the Docker host by updating the docker-compose.yml file as shown in Figure 24. Both options will require ISAIN to be restarted by executing

```
docker-compose down
docker-compose up -d
```

within the /opt/isain directory.

```
docker cp <path to new QOS file> isain_nifi_1:/opt/nifi/nifi-current/USER_QOS_PROFILES.xml
```

Figure 23: Running the mongoinport command line tool



```
nifi:
  image: 'uk-isain-proj01.rcnet.groupinfra.com:8082/isain/nifi:20200526.1'
  logging:
    driver: "json-file"
    options:
      max-file: "5"
      max-size: "10m"
  volumes:
    - /opt/isain/nifi:/home/nifi:z
    - /opt/isain/nifi/logs:/opt/nifi/nifi-current/logs:z
    - /opt/isain/nifi/database_repository:/opt/nifi/nifi-current/database_repository:z
    - /opt/isain/nifi/content_repository:/opt/nifi/nifi-current/content_repository:z
    - /opt/isain/nifi/flowfile_repository:/opt/nifi/nifi-current/flowfile_repository:z
    - /opt/isain/nifi/provenance_repository:/opt/nifi/nifi-current/provenance_repository:z
    - /opt/isain/nifi/state:/opt/nifi/nifi-current/state:z
    - /opt/isain/nifi/conf:/opt/nifi/nifi-current/conf:z
    - /opt/isain/nifi/lib_custom:/opt/nifi/nifi-current/lib_custom:z
    - /opt/isain/nifi/USER_QOS_PROFILES.xml:/opt/nifi/nifi-current/ USER_QOS_PROFILES.xml:z
  environment:
    HOME: '/home/nifi'
    NIFI_CLUSTER_IS_NODE: 'true'
    NIFI_CLUSTER_ADDRESS: 'isain-svr'
    NIFI_CLUSTER_NODE_PROTOCOL_PORT: '18083'
    NIFI_ZK_CONNECT_STRING: 'isain-svr:2181'
    NIFI_ELECTION_MAX_CANDIDATES: '1'
  network_mode: host
  hostname: "isain-svr"
  extra_hosts:
    - "isain-svr:0.0.0.0"
  depends_on:
    - apacheds
    - mongod
    - zookeeper
```

Figure 24: Mapping in a custom USER_QOS_PROFILES.xml


Refer to the ISAIN Developer Guide [\[1\]](#) for details on how to add a new DDS datatype, and the AdLink Opensplice Community Edition documentation [\[2\]](#) for how to configure the USER_QOS_PROFILES.xml file.

11 Real Time Troubleshooting

11.1 Overview

When integrating a new AI or application within ISAIN, it is likely that errors will occur. Therefore, it is important that a number of tools are available to monitor, analyse and move towards fixing these issues as they occur. This section will cover how to use both NiFi and Kibana to complete these tasks, utilising their functionality to build the best picture of the issues as possible.

11.2 NiFi

Out of the box, NiFi provides a selection of tools and techniques that can be used to monitor and find any issues within the flows and processors. The first of this is the Bulletin Board. This can be accessed by clicking on the menu icon  in the top-right of the NiFi display and selecting the 'Bulletin Board' option. This brings up the window shown in Figure 25.

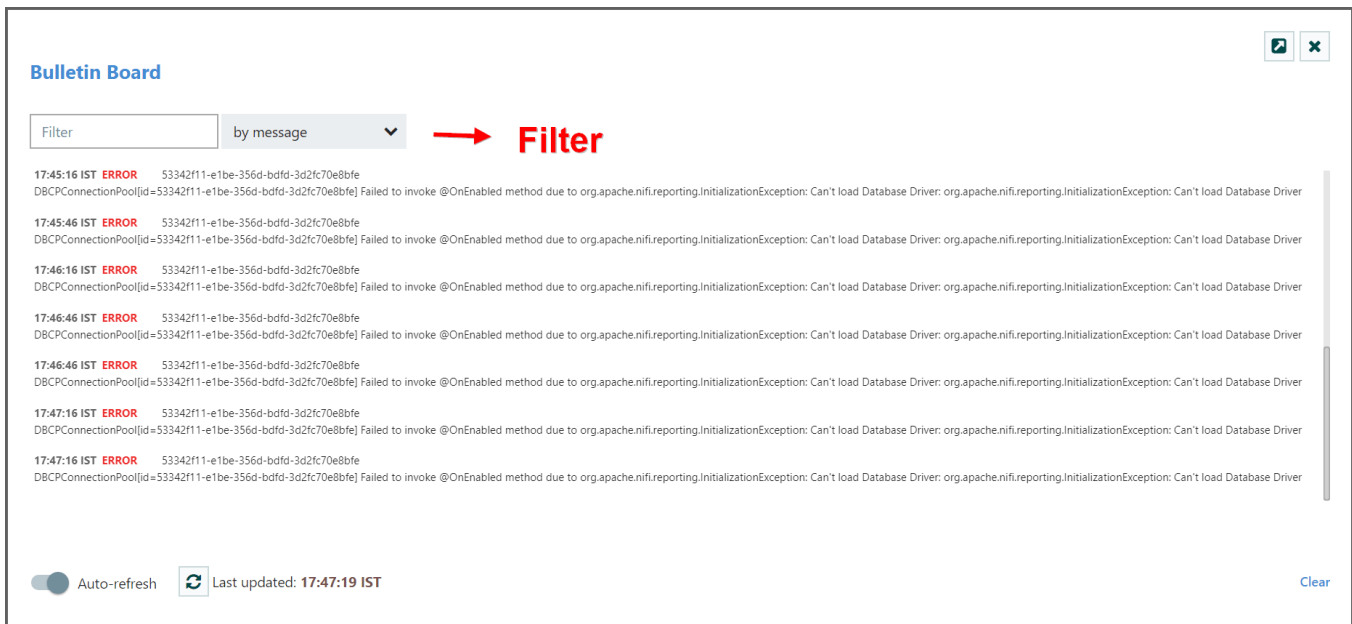

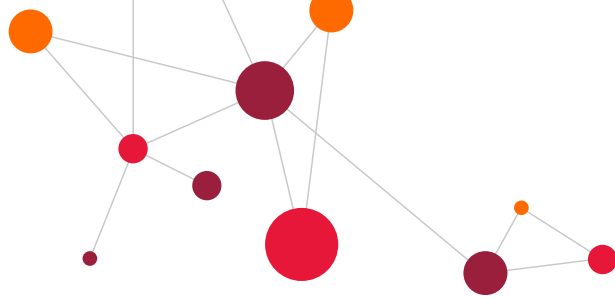


Figure 25: Apache NiFi Bulletin Board

Any errors that occur within the network are logged here, with the most recent appearing at the top of the list. If the network does not seem to be working correctly, but the location of the error is not clear, this is the best place to start, as it will display all error messages along with the location, allowing the issues to be found. Processors with errors, or Processor Groups which contain Processors with errors, are marked with the icon , clearly showing where the issue is. Hovering over the marker with the mouse pointer will display the error messages being produced.

Once the source of an issue has been identified, NiFi provides more tools to understand the data passing through that part of the network, and move towards identifying and fixing the issue, using the error message to understand even further. The first is being able to view the FlowFiles in queues. To do this, right click on the connection queue and select the 'List Queue' option as shown in Figure 26. A list of all the FlowFiles in the queue will be displayed, along with information about when they arrived, and access to the actual contents and attributes of the FlowFile.



This is particularly useful if the queue is backed up, as it can give an insight as to the contents of the data and why the processor may be rejecting them.

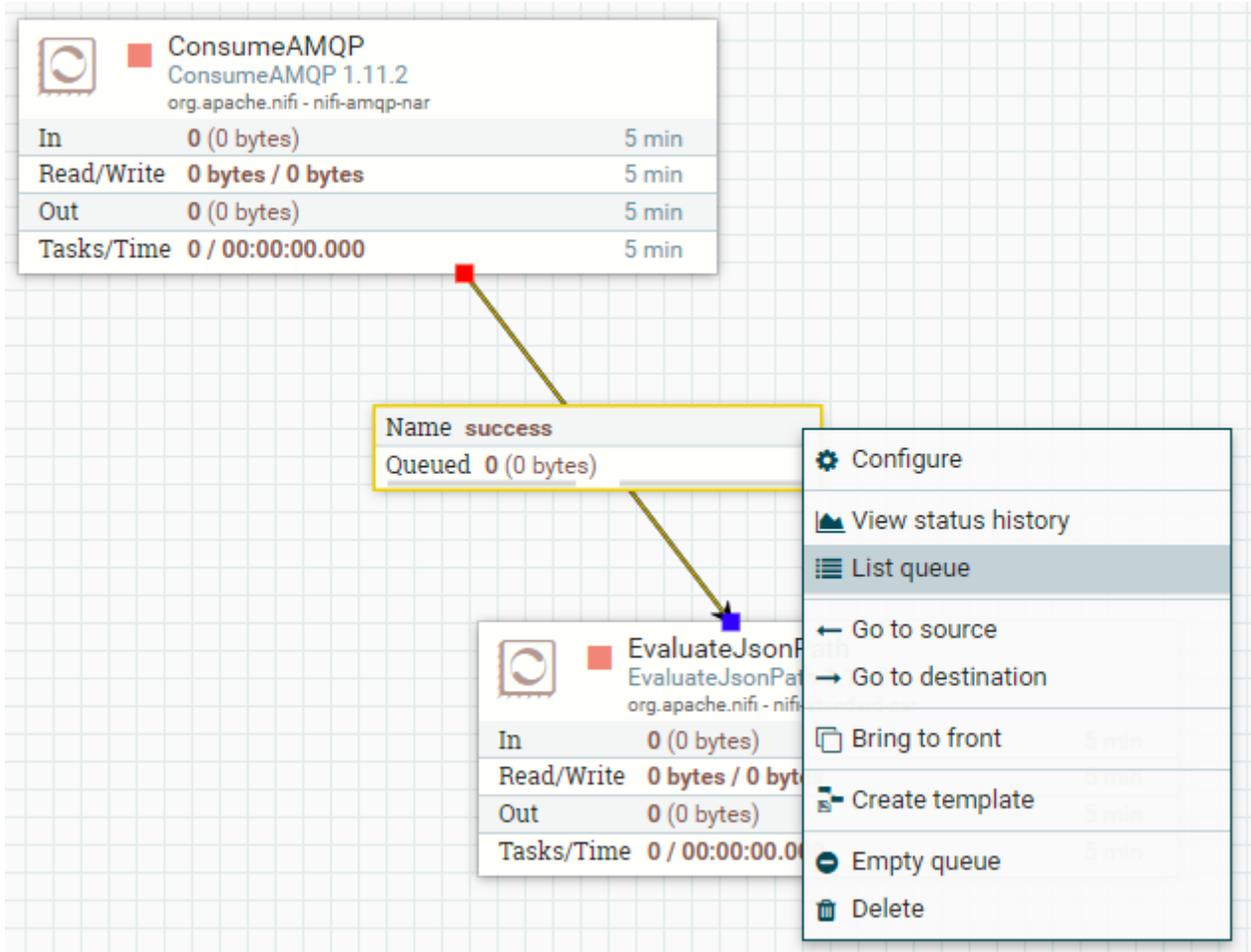


Figure 26: Viewing a NiFi connection queue

Another similar technique is viewing the Data Provenance of a processor. This can be achieved by right clicking on any processor and selecting the 'Data Provenance' option. This will show a list of events that have occurred within the processor, as shown in Figure 27.

NiFi Data Provenance

Displaying 193 of 193
Oldest event available: 07/18/2016 11:09:24 EDT


Filter by component n...

Showing the most recent events.


Date/Time	Type	FlowFile Uuid	Size	Component Name	Component Type
07/18/2016 11:09:57.574 EDT	CONTENT_MODIFIED	0fe67e3c-9408-4eff-8772-252f114b399f	2.91 KB	ExecuteSQL	ExecuteSQL
07/18/2016 11:10:04.224 EDT	ATTRIBUTES_MODIFIED	0fe67e3c-9408-4eff-8772-252f114b399f	2.91 KB	UpdateAttribute	UpdateAttribute
07/18/2016 11:10:13.713 EDT	CONTENT_MODIFIED	0fe67e3c-9408-4eff-8772-252f114b399f	5.76 KB	ConvertAvroToORC	ConvertAvroToORC
07/18/2016 11:10:40.352 EDT	SEND	0fe67e3c-9408-4eff-8772-252f114b399f	5.76 KB	PutHDFS	PutHDFS
07/18/2016 11:10:40.352 EDT	ATTRIBUTES_MODIFIED	0fe67e3c-9408-4eff-8772-252f114b399f	5.76 KB	PutHDFS	PutHDFS
07/18/2016 11:11:25.506 EDT	CONTENT_MODIFIED	0fe67e3c-9408-4eff-8772-252f114b399f	422 bytes	ReplaceText	ReplaceText
07/18/2016 11:11:48.435 EDT	SEND	0fe67e3c-9408-4eff-8772-252f114b399f	422 bytes	PutHiveQL	PutHiveQL
07/18/2016 11:11:48.871 EDT	DROP	0fe67e3c-9408-4eff-8772-252f114b399f	422 bytes	PutHiveQL	PutHiveQL
07/18/2016 11:09:57.541 EDT	CONTENT_MODIFIED	1b105de2-eb64-4ff1-b465-4fd8844da437	2.88 KB	ExecuteSQL	ExecuteSQL
07/18/2016 11:10:04.224 EDT	ATTRIBUTES_MODIFIED	1b105de2-eb64-4ff1-b465-4fd8844da437	2.88 KB	UpdateAttribute	UpdateAttribute
07/18/2016 11:10:13.657 EDT	CONTENT_MODIFIED	1b105de2-eb64-4ff1-b465-4fd8844da437	5.77 KB	ConvertAvroToORC	ConvertAvroToORC
07/18/2016 11:10:40.162 EDT	SEND	1b105de2-eb64-4ff1-b465-4fd8844da437	5.77 KB	PutHDFS	PutHDFS
07/18/2016 11:10:40.163 EDT	ATTRIBUTES_MODIFIED	1b105de2-eb64-4ff1-b465-4fd8844da437	5.77 KB	PutHDFS	PutHDFS
07/18/2016 11:11:25.505 EDT	CONTENT_MODIFIED	1b105de2-eb64-4ff1-b465-4fd8844da437	422 bytes	ReplaceText	ReplaceText
07/18/2016 11:11:46.354 EDT	SEND	1b105de2-eb64-4ff1-b465-4fd8844da437	422 bytes	PutHiveQL	PutHiveQL
07/18/2016 11:11:48.871 EDT	DROP	1b105de2-eb64-4ff1-b465-4fd8844da437	422 bytes	PutHiveQL	PutHiveQL
07/18/2016 11:09:57.567 EDT	CONTENT_MODIFIED	213f984e-e355-42a6-8632-78e1bc37bacf	2.9 KB	ExecuteSQL	ExecuteSQL
07/18/2016 11:10:04.224 EDT	ATTRIBUTES_MODIFIED	213f984e-e355-42a6-8632-78e1bc37bacf	2.9 KB	UpdateAttribute	UpdateAttribute
07/18/2016 11:10:13.699 EDT	CONTENT_MODIFIED	213f984e-e355-42a6-8632-78e1bc37bacf	5.76 KB	ConvertAvroToORC	ConvertAvroToORC

Last updated: 11:28:49 EDT

Figure 27: Apache NiFi Data Provenance display

By clicking the  icon on the left of each row, a new window will open displaying all the data associated with the event, including the FlowFile attributes with modified attributes highlighted, and its content, which can be viewed before and after the processors execution. This provides clear picture of what the processor has done at that particular point in time. This tool is invaluable in understanding what the flow is doing at each stage, and can be used to find and fix issues as they appear.

11.3 Kibana

Kibana forms part of Elastic Stack, a tool used to collate entries from a number of disparate log files and store them in a single location to enable easy searching and viewing of log entries. Kibana is an analysis and visualisation tool used to display these logs in an understandable but informative manner, and as such it plays an important role in the real time troubleshooting of the network. Kibana can be accessed via the Console, by selecting the 'Kibana' option from the menu  in the top left of the console, and provides a number of ways to narrow down the data available. Kibana is a more suitable troubleshooting tool when the perceived issue exists in one of the components of the network, not the NiFi network itself, where NiFi's tools are more suitable.

Within Kibana, navigate to the 'Discover' page. If Kibana says that no indexes exist, follow the instructions provided to create one. If using the default setup, then the matching pattern is best set as 'filebeat-*'. Refer to the Kibana documentation for further instructions using the URL in Table 2. This page looks like Figure 28, and is the best place to begin troubleshooting issues.



Figure 28: Kibana Discover page

Along the top of the screen is a search bar, where custom queries can be written and computed, and a time filter, allowing logs for a particular period of time to be retrieved. Along the left hand side of the page are a number of fields that can be used to filter the log entries. These filters can be built up to filter the data more narrowly. The list of log entries that match these criteria are displayed in the centre of the screen, with a graph showing the frequency of entries at different periods of time.

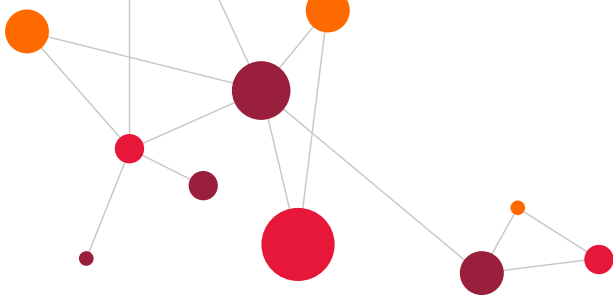
The search bar can be used to make plain-text searches, which can be used to find log entries that match a particular query. Below the search bar, filters can be added on any of the number of attributes available. The 'application' field added earlier is especially useful to filter on when troubleshooting. The 'Available fields' column along the left-hand side of the page can be used to change what is shown. For example, selecting and adding the 'message' field will only show the message in the display, with no other data cluttering the view. This option is also very useful for troubleshooting, as it allows a clear view of the message.

11.4 Resetting the network

It can be useful to clear all the logs and data currently stored in the network between runs. This can be achieved by deleting certain directories/files containing logs and other data that may cause unexpected behaviour in the next run.

The first step is to stop all the containers currently running, this is done by running the 'docker-compose down' command in /opt/isain. After the network has been stopped, delete all the logs found in these locations:

- /opt/isain/console/logs/
- /opt/isain/nifi/logs/
- /opt/isain/zookeeper/logs/
- /opt/isain/zookeeper/data/
- /opt/isain/mongodb/logs/

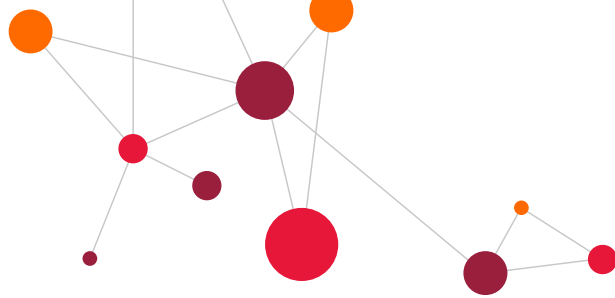


- /opt/isain/elk/elasticsearch/logs/
- /opt/isain/elk/logstash/logs/
- /opt/isain/elk/kibana/logs/


To remove all the provenance and FlowFile data from NiFi, delete these directories as well:

- /opt/isain/nifi/flowfile_repository
- /opt/isain/nifi/provenance_repository
- /opt/isain/nifi/content_repository
- /opt/isain/nifi/database_repository

Once all these directories/files have been deleted, the network can be restarted and will not contain any data from any previous runs. Steps taken before may have to be taken again, like re-importing the NiFi template, or adding indexes to Kibana, so make sure any relevant backups are made and saved before clearing this data.



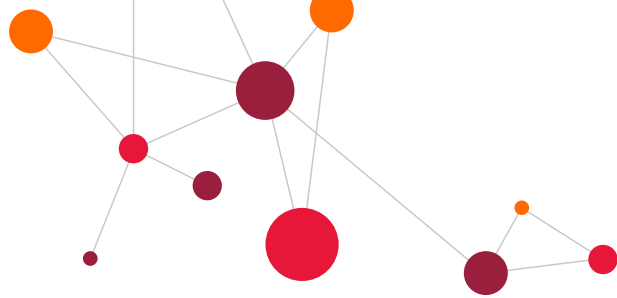
12 Post Scenario analysis

After an experiment/scenario is completed, it is important that the results can be analysed. To achieve this, the same techniques used in the previous section on real time troubleshooting can be used. NiFi's Data Provenance view can be used to understand the actions of a particular processor, possibly uncovering issues that led to unexpected behaviour throughout the run, as well as simply analysing how the network handled the inputs and outputs from the Console, AIs and Synthetic Environments. The route these events made throughout the network can be viewed using the  icon on the provenance event, which displays a graph of the route that particular FlowFile followed, both prior to and after it arrived at the current processor. This can be invaluable in understanding and analysing how the network handles different types of data.

The same can be said for Kibana. The techniques described previously can be used to collate and analyse the logs of all the components of the network, allowing an understanding of each components actions to be made. The Console Event Log can be used in tandem with Kibana to understand how AI Responses were handled during the scenario. The entries in the Console Event Log can also be found in Kibana, as they are written to the NiFi log file, and the log files entries contain more information about the event, such as all the attributes associated with the response or interaction, so Kibana should be used in place for the Console Event Log if a deeper understanding is required.

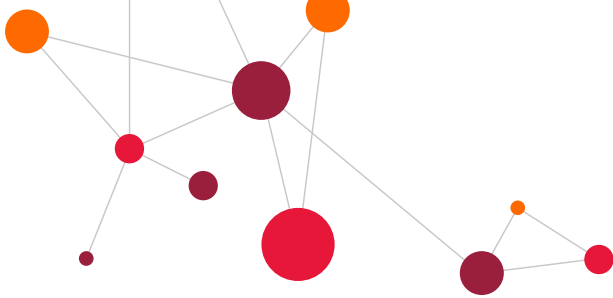
APPENDICES





Appendix A - Glossary

Acronym	Description
AI	Artificial Intelligence
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
DDS	Data Distribution Service
Dstl	Defence Science and Technology Laboratory
HITL	Human In The Loop
HOOTL	Human Out Of The Loop
HOTL	Human On The Loop
ISAIN	Intelligent Ship Artificial Intelligence Network
JMS	Java Message Service
JSON	JavaScript Object Notation
LDAP	Lightweight Directory Access Protocol
OACS	Open Architecture Combat System
REST	REpresentational State Transfer
SADM	Ship Air Defence Model
UI	User Interface
URL	Uniform Resource Locator



Appendix B – Example LDIF File

```
# File rusers.ldif

dn: cn=AW01,ou=isainUsers,o=dst1
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
cn: AW01
description: Air Warfare Officer account one
sn: AW01
mail: AW01@dst1
userpassword: AW01password

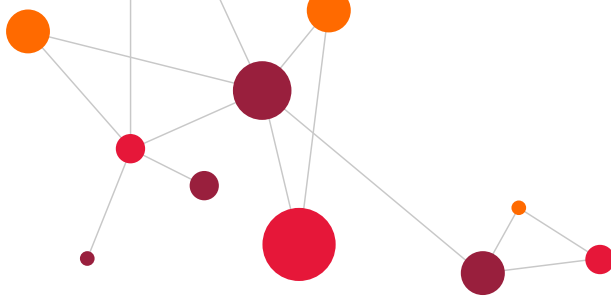
dn: cn=AW02,ou=isainUsers,o=dst1
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
cn: AW02
description: Air Warfare Officer account two
sn: AW02
mail: AW02@dst1
userpassword: AW02password

dn: cn=PW01,ou=isainUsers,o=dst1
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
cn: PW01
description: Principal Warfare Officer account one
sn: PW01
mail: PW01@dst1
userpassword: PW01password

dn: cn=PW02,ou=isainUsers,o=dst1
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
cn: PW02
description: Principal Warfare Officer account two
sn: PW02
mail: PW02@dst1
userpassword: PW02password

dn: cn=C01,ou=isainUsers,o=dst1
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
cn: C01
description: Commanding Officer account two
sn: C01
mail: C01@dst1
userpassword: C01password

dn: cn=TPD(A)1,ou=isainUsers,o=dst1
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
cn: TPD(A)1
description: Air Tactical Picture Director account one
```

```
sn: TPD(A)1
mail: TPD(A)1@dstl
userpassword: TPD(A)1password

dn: cn=APS1,ou=isainUsers,o=dstl
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
cn: APS1
description: Air Picture Supervisor account one
sn: APS1
mail: APS1@dstl
userpassword: APS1password

dn: cn=EWD1,ou=isainUsers,o=dstl
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
cn: EWD1
description: Electronic Warfare Director account one
sn: EWD1
mail: EWD1@dstl
userpassword: EWD1password

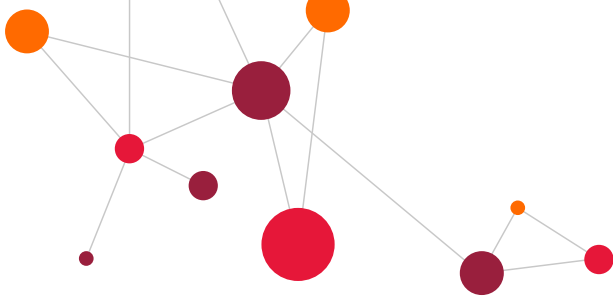
dn: cn=APS2,ou=isainUsers,o=dstl
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
cn: APS2
description: Air Picture Supervisor account two
sn: APS2
mail: APS2@dstl
userpassword: APS2password

dn: cn=EWD2,ou=isainUsers,o=dstl
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
cn: EWD2
description: Electronic Warfare Director account two
sn: EWD2
mail: EWD2@dstl
userpassword: EWD2password

dn: cn=AWO,ou=isainRoles,o=dstl
objectclass: top
objectclass: groupofnames
cn: AWO
description: Air Warfare Officers
member: cn=AWO1,ou=isainUsers,o=dstl
member: cn=AWO2,ou=isainUsers,o=dstl

dn: cn=PWO,ou=isainRoles,o=dstl
objectclass: top
objectclass: groupofnames
cn: PWO
description: Principal Warfare Officers
member: cn=PWO1,ou=isainUsers,o=dstl
member: cn=PWO2,ou=isainUsers,o=dstl

dn: cn=APS,ou=isainRoles,o=dstl
objectclass: top
objectclass: groupofnames
cn: APS
description: Air Picture Supervisors
```



```
member: cn=AP51,ou=isainUsers,o=dstl
member: cn=AP52,ou=isainUsers,o=dstl

dn: cn=EWD,ou=isainRoles,o=dstl
objectclass: top
objectclass: groupofnames
cn: EWD
description: Electonic Warfare Directors
member: cn=EWD1,ou=isainUsers,o=dstl
member: cn=EWD2,ou=isainUsers,o=dstl

dn: cn=officers,ou=isainRoles,o=dstl
objectclass: top
objectclass: groupofnames
cn: officers
description: officer personnel
member: cn=PWO,ou=isainRoles,o=dstl
member: cn=AWO,ou=isainRoles,o=dstl
member: cn=C01,ou=isainUsers,o=dstl

dn: cn=airdomain,ou=isainRoles,o=dstl
objectclass: top
objectclass: groupofnames
cn: airdomain
description: operators in the air domain
member: cn=AWO,ou=isainRoles,o=dstl
member: cn=PWO,ou=isainRoles,o=dstl
member: cn=C01,ou=isainUsers,o=dstl
member: cn=TPD(A)1,ou=isainUsers,o=dstl
member: cn=APS,ou=isainRoles,o=dstl
member: cn=EWD,ou=isainRoles,o=dstl

dn: cn=surfacedomain,ou=isainRoles,o=dstl
objectclass: top
objectclass: groupofnames
cn: surfacedomain
description: operators in the surface domain
member: cn=PWO,ou=isainRoles,o=dstl
member: cn=C01,ou=isainUsers,o=dstl

dn: cn=subsurfacedomain,ou=isainRoles,o=dstl
objectclass: top
objectclass: groupofnames
cn: subsurfacedomain
description: operators in the sub-surface domain
member: cn=PWO,ou=isainRoles,o=dstl
member: cn=C01,ou=isainUsers,o=dstl
```