

DARPA Documentation and Decision Support Sprint
Interface Control Document
Qualification Task

October 2026
Version 1.1



Defense Advanced Research Projects Agency
Biological Technologies Office
675 North Randolph Street
Arlington, VA 22203-2114
D2Sprint@darpa.mil

Distribution Statement ‘A’ (Approved for Public Release, Distribution Unlimited)

1 Contents

1 CONTENTS 3

2 INTRODUCTION 4

3 SUBMISSION REQUIREMENTS 4

3.1 INPUT DATA INTERFACE 5
3.2 OUTPUT DATA INTERFACE 5
3.3 REQUIRED DOCKER COMMAND 6
3.4 RUNTIME AND RESOURCE CONSTRAINTS 7
3.5 CONTAINER SCANNING 7

4 SUBMISSION PROCEDURE 7

4.1 HOW TO USE THE CONTAINER REGISTRY 7
4.2 OUTPUT VALIDATION SCRIPT 8

5 EVALUATION ENVIRONMENT 9

6 APPENDIX A – OUTPUT JSON SCHEMA METADATA 9

2 Introduction

The goal of the Documentation and Decision support Sprint (D2) is to catalyze advancements in combat medicine through targeted prize incentives, specifically seeking autonomous solutions that enable hands-free DD Form 1380 documentation and Tactical Combat Casualty Care (TCCC)-aligned decision support.

The D2 will establish the current State of the Art (SOA) for autonomous TCCC documentation and decision support tools, assessing gaps in existing datasets of TCCC, and mature assessment metrics leading into larger research efforts to develop more mature and applicable documentation and decision support tools for the Warfighter. The competition will be structured into two distinct lanes:

- **Lane 1—Documentation:** Participants will develop algorithms to autonomously and accurately populate a digital DD Form 1380 (TCCC Card) using passively collected data (e.g., audio, video, accelerometry).
- **Lane 2—Decision Support:** Participants will create algorithms to provide accurate, real-time clinical decision support to medics based on established Committee on Tactical Combat Casualty Care (CoTCCC) Guidelines and Joint Trauma System Clinical Practice Guidelines (JTS CPGs).

This Interface Control Document (ICD) defines the technical, data, execution, hardware, and submission requirements for team solutions submitted to the D2 Qualification Task.

For the event qualification, each team shall build and push a Docker container to the designated container registry. The submitted container shall ingest image data, perform injury classification predictions, and produce a structured JavaScript Object Notation (JSON) output that conforms to the interface defined in this document.

3 Submission Requirements

Each potential competitor team will:

1. Build a Docker-compatible container image.
2. Include all required source code, model files, dependencies, and runtime assets within the image.
3. Push the image to the designated container registry.
4. Provide the submission information to JHU/APL.
5. Ensure the container can execute without access to the public internet.
6. Ensure the container complies with the resource constraints defined in this document.

A representative container structure may include:

```
/app/
 ├── inference.py
 ├── entrypoint.sh
 ├── requirements.txt
 ├── model/
|   └── weights/
```

```
├── input/
└── output/
```

The internal structure is not mandatory, provided the container conforms to the required runtime and input/output interfaces.

3.1 Input data interface

The container shall accept image files located in:

```
/app/input
```

The input directory will use a flat directory structure with the folder containing all images for processing.

The evaluation environment may provide one or more of the following formats:

- .jpg
- .jpeg
- .png

The qualification test set is expected to be approximately 20% the size of the training set and will consist entirely of held-out images

3.2 Output data interface

The container shall write exactly one primary results file:

```
/app/output/predictions.json
```

The output file shall:

- Contain valid JSON
- Use UTF-8 encoding
- Use double quotes for property names and string values
- Contain no NaN, Infinity, or nonstandard JSON values
- Conform to the schema defined in this document
- Include exactly one prediction for each of four required body sites for every input image (a body site is comprised of fields “body_region” and “laterality”)
    - body_region: upper_extremity; laterality: left
    - body_region: upper_extremity; laterality: right
    - body_region: lower_extremity; laterality: left
    - body_region: lower_extremity; laterality: right
- Use one of the following predicted classes for the “injury_type” field:
    - “no_injury”
    - “wound”
    - “amputation”
    - “not_testable”

3.2.1 Output JSON Schema

The output must conform to the following JSON schema:

```
{
  "schema_version": "1.0",
  "submission": {
    "team_name": "example-compliant-submission",
    "version": "1.0.0",
    "email": "team@gmail.com"
  },
  "predictions": [
    {
      "image_id": "image_000001.jpg",
      "sites": [
        {
          "body_region": "upper_extremity",
          "laterality": "left",
          "injury_type": "wound"
        },
        {
          "body_region": "upper_extremity",
          "laterality": "right",
          "injury_type": "no_injury"
        },
        {
          "body_region": "lower_extremity",
          "laterality": "left",
          "injury_type": "amputation"
        },
        {
          "body_region": "lower_extremity",
          "laterality": "right",
          "injury_type": "not_testable"
        }
      ]
    }
  ]
}
```

3.3 Required docker command

JHU/APL will execute team containers using a command equivalent to:

```
docker run --rm \
  --gpus all \
  --network none \
  -v /input:/app/input:ro \
```

```
  -v /output:/app/output:rw \
  team_image:<image_digest>
```

The exact registry path and image digest will be provided by the team.

3.4 Runtime and Resource Constraints

Each submission shall conform to the following limits:

Resource	Maximum Allowed
GPUs	1 GPU
GPU memory	Less than 16 GB
CPU	8 CPU cores
System RAM	Less than 32 GB
Runtime disk usage	Less than 20 GB
Compressed or stored container image size	Less than 20 GB
Evaluation runtime	1 hour

3.5 Container Scanning

Each submission must pass a ClamAV malware scan. To minimize pipeline rejection, teams are encouraged to scan their container filesystems locally before submitting. The official ClamAV Docker image can be used to scan a directory containing the exported container filesystem.

Export your container filesystem:

```
docker save team_image | tar -xC ./extracted-container
```

Run the ClamAV scan:

```
docker run --rm \
            -v $(pwd)/extracted-container:/scandir \
            clamav/clamav:latest clamscan -r /scandir
```

4 Submission Procedure

Each team shall create a free GitLab account and a private GitLab project. The team shall push their submission container to the project's GitLab Container Registry and grant the JHU/APL evaluation account (@D2-evaluator) ***Reporter*** access to the project. JHU/APL will pull the immutable image digest from the private registry for evaluation.

4.1 How to use the container registry

The steps for the team to follow are outlined below:

1. Create a free Gitlab account.
2. Create a private project.

3. Add the JHU/APL evaluation account.
    a. Open the project
    b. Go to Manage → Members
    c. Invite @D2-evaluator as ***Reporter***
4. Build the docker image. Use the name `<team-name>-d2-qualification-task`

```
docker build -t <team-name>-d2-qualification-task:<version_tag> .
```

5. Authenticate to Gitlab.

```
docker login registry.gitlab.com
```

6. Enter your username and password or Personal Access Token.
7. Tag the image.

```
docker tag <team-name>-d2-qualification-task:<version_tag> \
        registry.gitlab.com/<project_name>/<team-name>-d2-qualification-task:<version_tag>
```

8. Push the image.

```
docker push registry.gitlab.com/<project_name>/<team-name>-d2-qualification-task:<version_tag>
```

9. Record the digest.

```
docker inspect --format='{{index .RepoDigests 0}}' \
registry.gitlab.com/<project_name>/<team-name>-d2-qualification-task:<version_tag>
```

10. Submit the following information via email to APL-D2C-Help@jhuapl.edu:
    - Team Name
    - GitLab Username
    - Project URL
    - Container Image
    - Image Tag
    - Image Digest
    - Technical Contact

Under a free Gitlab account, there is a maximum of 5 users that can be on a project. Team members should keep track of this when inviting other members of their team to the project. One spot must be left for the JHU/APL evaluation account to join. Teams may also use their Gitlab project to store their project's source code, but it is not required.

4.2 Output Validation Script

JHU/APL will provide teams with a test script to validate their output:

- Uses valid JSON
- Contains all required fields
- Uses valid enumerated values
- Includes all required image and site records
- Conforms to the required schema

Teams will be given access to this script before submitting their qualification image.

Example validation workflow:

```
python3 validate_predications.py <PATH_TO_PREDICTIONS_JSON>
```

If the predictions file is valid, the script prints a success message and returns exit code 0. If the script encounters an invalid JSON syntax or a schema violation, it catches the structural flaws, missing metrics, or syntax errors, lists every individual violation path (such as $.predictions[0].sites[2].injury_type), and returns an exit code of either 1 or 2.

5 Evaluation Environment

JHU/APL will run each submitted container on one of the following NVIDIA GPUs:

- NVIDIA A40
- NVIDIA L40
- NVIDIA H100

The evaluation environment will use:

Component	Evaluation Configuration
NVIDIA driver	Version 580 or later
CUDA compatibility	CUDA 13 or earlier
Maximum GPUs	1
CPU architecture	x86-64
Operating system	Linux
Container runtime	Docker with NVIDIA Container Toolkit

Submissions shall be capable of executing on each listed GPU type without requiring image modification or rebuilding. Teams shall not assume that a specific GPU model will be assigned. Team containers should contain the CUDA runtime and libraries, but should not install or package an NVIDIA kernel driver.

6 Appendix A – Output JSON Schema Metadata

Submission metadata requirements:

Field	Type	Required	Description
schema_version	String	Yes	Version of the output interface schema
submission.team_name	String	Yes	Official team or team name
submission.version	String	Yes	Team-defined submission version
submission.email	String	Yes	Primary technical contact email

Prediction requirements:

Field	Type	Required	Allowed Values or Constraints
image_id	String	Yes	Exact input image file name

sites	Array	Yes	Must contain exactly four site predictions
body_region	String	Yes	upper_extremity, lower_extremity
laterality	String	Yes	left, right
injury_type	String	Yes	amputation, wound, no_injury, not_testable