Interface Control Document
Phase 3
Data Competition

Revision 03
May 2025

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

1 Contents

2 Introduction

The goal of this document is to convey the high-level concept and infrastructure used to evaluate team submissions in the Data Competition. This document also describes the requirements and interface for successful integration with JHU/APL's evaluation system. This document is specific to the Data Competition for Phase 3. For information on the Systems and Virtual Competitions please refer to their respective *Interface Control Documents*. Significant revisions from past versions in this document during the current phase are indicated by blue text.

The remainder of this document is organized as follows: Section 3 describes the evaluation system, messaging protocol, and the evaluation process; Section 4 describes the evaluation environment for Phase 2; Section 5 describes requirements and resources for submission preparation; and Section 6 describes the procedure for submission. Appendices provide supporting information.

3 Evaluation System

This section contains information about the evaluation system planned for workshop and challenge events in the Data Competition. All formal evaluation procedures will be performed on JHU/APL networks.

Model submissions will be evaluated using a *held-out* test dataset in a simulated prediction environment, in accordance with the scoring procedure described in the DTC Rules Document. Models will be evaluated on three prediction tasks: First Look (Task 1), Continuous Alert (Task 2), and Resource Allocation (Task 3). For Task 1 and Task 2, models will be evaluated on a single patient case at a time; for Task 3, models will be evaluated on a collection of patient cases. In each task, models are provided with input data and provide intervention predictions as output. Input data is provided in sequential order (where applicable) with relative timestamps when available. Input data includes a combination of tabular data, casualty descriptions, and physiological signals. Models will be responsible for accumulating or storing past data within a case, if necessary. Input and output format differs between tasks, as described in detail below.

Segmented training datasets will be provided in the same form as data will appear during the evaluation. See the DTC Forum for updates to the training datasets available on AWS.

Figure 1 provides a high-level description of the three interacting modules in the JHU/APL evaluation system:

1. **Evaluator**, which hosts all logic required to distribute test data to the teams' client containers (via Rabbit MQ) and evaluate their responses;
2. **Rabbit MQ Server**, which hosts the server that governs communication between the Client Container and the Evaluator;
3. **Client Container**, a Docker container which processes input data, runs model inference, and responds with LSI predictions (via Rabbit MQ). Also referred to simply as the Client.

For each competition event, teams are expected to provide code for building the Client Container (see Section 5 for development resources). After the submission deadline, AWS administrators will pull submitted code and build the Client Container (see Section 6 for details on submission

procedure). This Client Container will then run alongside the Evaluator within the JHU/APL network. During evaluation, the Evaluator sends input data to the Client Container and receives prediction responses via Rabbit MQ's messaging protocol (see Section 3 for more information about the evaluation process and message formats). At the end of the evaluation, the system produces reports with Client responses and performance metrics for each team.

The Rabbit MQ Server and Evaluator will be developed and maintained by JHU/APL. Software submissions must comply with the JHU/APL evaluation system in order to be evaluated. JHU/APL will also provide a Client Shell as a starting point for submission preparation (see Section 5.2). To confirm compliance with the evaluation system, the JHU/APL team will provide a Continuous Integration/Continuous Development (CI/CD) system integrated with the DTC AWS network and teams' code repositories that will build provided code and perform an abbreviated evaluation run to test submission compliance (see Section 5.4). Successful test outcome from the CI/CD indicates that the submission is compliant with the evaluation system. Teams will then need to follow the submission procedure to provide code for evaluation (see Section 6).

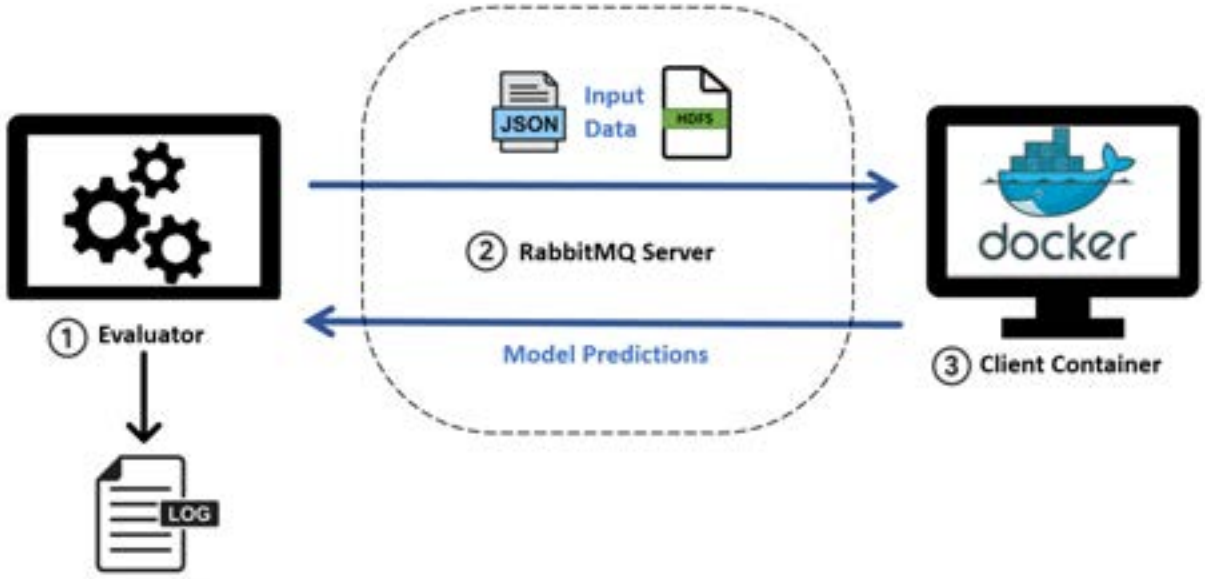

Figure 1: High-level architecture of the JHU/APL evaluation system. *This system is composed of 1) the Evaluator, 2) Rabbit MQ server, and 3) the Client Container, containing the team's model. The Evaluator sends test data to the Client Container using RabbitMQ messages, and the Client Container responds with the LSI predictions.*

In Section 3.1, we describe the main responsibilities of the Evaluator and Client Container and the progression of messages passed between these two components during an evaluation run, as well as the results expected after evaluation. In Section 3.2, we describe the message formats expected between Evaluator and Client Container.

3.1 Evaluation Process

The Evaluator is responsible for accessing the held-out test dataset, packaging and serving data segments to the client container via RabbitMQ, and logging responses from the Client Container.

The Client Container is responsible for housing the submitted model, ingesting data provided by the Evaluator via RabbitMQ, aggregating or storing information across data segments (as needed), preprocessing data and running model inference, and sending model responses with LSI predictions back to the Evaluator.

After the Evaluator establishes connection with the Client Container, the Evaluator will begin sending data segments to the Client Container to perform model inferences and return LSI predictions. Evaluation occurs one case at a time with the Evaluator sending data segments within each case in temporal order from beginning to end.

The Evaluator and Client communicate using the following message formats (described in detail in Section 3.2):

- *Predict* message, containing a data segment provided to the Client for predicting LSIs.
- *Response* message, containing LSI predictions from the Client based off of data received
- *Acknowledge* message, containing status information about evaluation progress
- *Timed Out* message, indicating to the Client that the prediction process timed out
- *Error* message, indicating to the Client that the prediction response was mal-formed
- *Cleanup* message, indicating to the Client that the case is complete and evaluation is continuing with the next case

The following steps describe the sequence of messages sent between the Evaluator and the Client Container during an evaluation run:

1) **Evaluator establishes connection to the Client Container.** An initial message is sent from the Evaluator to the Client Container to establish the RabbitMQ channel for the evaluation run.
2) **Evaluator sends Predict message.** The Evaluator sends a new data segment in a Predict message and waits for a Response message from the Client.
3) **Client performs inference.** The Client receives the Predict message, performs any preprocessing activities and model inference.
4) **Client sends Response message.** The Client sends a response containing the list of predicted LSIs (if any).
5) **Evaluator waits for Response message and saves output.** The Evaluator saves the Response message from the Client for calculating metrics. If the Response message is not received from the Client before the segment duration has elapsed, go to Step 6; if the Response is timely but not well-formed, go to Step 7; otherwise, go to Step 8.
6) **(If applicable) Evaluator sends a Timed Out message and resumes with next case.** If the Client does not produce Response message within time limit (segment duration), then the Evaluator sends a Timed Out message to the Client. The evaluation is then interrupted and resumes at Step 1, starting with the next case in the test dataset.
7) **(If applicable) Evaluator sends an Error message and resumes with next case.** If the Client does not produce a well-formed Response message (e.g., LSI label is misspelled), then the Evaluator sends an Error message to the Client. The evaluation is then interrupted and resumes at Step 1, starting with the next case in the test dataset.

8) **Evaluator sends Acknowledge message.** The Evaluator acknowledges receipt of a well-formed and timely Response message from the Client with status information about evaluation progress.
9) **Repeat Steps 2-8 for the remaining segments in the case (if applicable).** The Evaluator continues sending incremental segments of the data for the current case. The Client Container is responsible for storing historical data from previous segments within the same case. For the last segment of the case, the Predict message will indicate the end-of-case using a boolean flag, indicating the next Predict message will come from a new case.
10) **Evaluator confirms end of case with a Cleanup message.** After the prediction from the last segment of a case is received, the Evaluator sends a "clean-up" message to cue the Client to execute any process before beginning a new case.
11) **Repeat Steps 2-10 for remaining cases.** The Evaluator continues until all cases have been evaluated.

3.1.1 Response log

With each well-formed and timely response from the Client Container (Step 5 above), the following information will be stored to facilitate post-hoc metrics calculation and scoring:

- Patient case identifier
- Segment identifier
- Time window of evaluation segment
- Time elapsed from message sent to response received
- LSI predictions
- Optional response fields (e.g., embeddings, see Section 3.1.7)

This log will be provided to teams after each workshop and challenge event.

3.1.2 Metrics log

After an evaluation run, metrics will be computed from the Client response logs and the test dataset ground truth. This log will include scoring metrics by task, as described in the Rules Document.

3.2 Communication Protocol: Task 1 (First Look)

In this section, we describe the communications between the Evaluator and Client for Task 1 (First Look), where message types include: *Predict*, *Response, Acknowledge*, *Timed Out*, *Cleanup*, and *Error*.

3.2.1 Predict Message Format

For the First Look task, there is a single Predict message sent by the Evaluator to the Client for each case within an evaluation run. The Predict message contains one or more of the following data elements: simulated casualty report, hospital admission time, EHR data (including previous interventions), VS data, and metadata. Here is an example of the Predict message format for a single case:

```
{
```

```
 "segment_id": "OUmJvhCVC5",
 "case_id": "wv9ufeR87l",
 "end_of_case": True,
 "start_time": 0,
 "stop_time": 600,
 "hosp_adm_time": 1620,
 "num_bins": 5,
 "casualty_report": <dict>,
 "ehr": <dict>,
 "vs": <dict>
}
```

Message 1: Predict message format

Definitions for Predict message fields are as follows:

- *segment_id* is a unique string identifier for the current segment
- *case_id* is a unique string identifier for the current case, composed of many segments
- *end_of_case* is a boolean indicator that this is the last segment for the current case
- *start_time* is the start of the vitals data segment as time elapsed from start of case in seconds
- *stop_time* is the end of the vitals data segment as time elapsed from start of case in seconds
- *num_bins* is the vector length expected in the response message (see Section 3.2.2)
- *hosp_adm_time* is the time of hospital admission relative to the start of case in seconds
- *casualty_report* is a simulated casualty report similar to the format used for autonomous sensing for medical triage in the Systems Competition (new in Phase 3, details below)
- *ehr* is a dictionary containing Electronic Health Record (EHR) data, described below
- *vs* is a dictionary containing Vital Signs (VS) data, described below

Depending on data availability within each case and run, the *casualty_report, ehr,* and/or *vs* fields may be empty dictionaries or *None*. If present, all fields within these dictionaries are optional. For any available data, field names are consistent with the data dictionary and documentation provided with the unsegmented training dataset.

The provided *num_bins* contains the number of 15 minutes time-bins for predicting LSI events for the current case that span from the end of the segment (*stop_time*) up to and not exceeding 4 hours after hospital admission, where hospital admission is provided in *hosp_adm_time*.

In response to the Predict message, the Client is expected to respond with a Response message (see Section 3.2.2).

3.2.1.1 ehr Data Field

The *ehr* dictionary contains basic EHR data available upon "first look". Data contained in the EHR data may include: interventions received (both within the list of target scored LSIs and other LSIs), demographics, mechanism of injury, and initial assessment and diagnostic information. In Runs 1 and 2, only previous interventions are provided in basic EHR data; all basic EHR data is provided in Run 3. See Appendix A for list of fields included in the EHR data by run. See the Data Competition Rules Document for more information about evaluation runs.

Note that a large portion of the EHR data provided in training datasets will be excluded completely from the evaluation (for example, non-LSI procedures or medications, fields related to outcome or information not available in the acute period of treatment).

3.2.1.2 Casualty Report Field

The Casualty Report is intended to mimic the information that would be captured by autonomous, stand-off sensing systems such as those being developed in the Systems Competition. Information from the EHR record was mapped to Casualty Report fields according to definitions provided in the Phase 3 Systems competition Rules. While the intention is to provide correct information in the Casualty Reports, they should be treated as noisy inputs with no guarantee of accuracy or completeness.

3.2.1.3 vs Data Field

The *vs* dictionary contains Vital Signs (VS) data from the beginning of the case. Timeseries data will have timestamps relative to the start of the case (in seconds). The *vs* dictionary will include any available trends and signal data from the first 5 minutes of pre-hospital VS data with an internal structure that mirrors the file structure provided in the training dataset.

Here is an example of the *vs* dictionary containing all VS data sources:

```
{
 "signal": <dict>,
 "trends": <dict>
}
```

All signal and trends fields contain dictionaries that match the structure and field names of their respective HDF5 files in the training dataset and the accompanying dataset documentation. Only available data will be included, so all fields are optional.

3.2.2 Response Message Format

For Task 1, Clients are expected to respond to the Predict message with LSI predictions over time represented as a dictionary with key-value pairs of each LSI label and a vector of confidence scores by time bin, along with a single confidence score that the patient receives any LSI at any time up to 4 hours after hospital admission ("any_lsi). Each confidence score must be a value between 0 and 1, inclusive, with no other normalization required across predictions for the same LSI or across LSIs (i.e., confidence scores need not sum to 1). The dictionary must contain a key for each of the possible LSI labels plus "any_lsi" (see Message 2). With exception of "any_lsi", all prediction vectors must have length equal to "num_bins" provided in the Predict message. Prediction vectors that are shorter or longer will result in a non-compliant submission prediction response.

Here is an example Response message:

```
{
 "segment_id": "OUmJvhCVC5",
 "lsi_predictions":
```

```
 {
  "airway_invasive": [0.2746, 0.8912, ..., 0.0437, 0.6621],
  "blood_products": [0.5089, 0.1174, ..., 0.9346, 0.2805],
  "chest_decompression": [0.7613, 0.4028, ..., 0.0951, 0.8479],
  "surgical": [0.9421, 0.2365, ..., 0.7810, 0.3996],
  "vaso_cardioactive_medications": [0.0543, 0.8159, ..., 0.6284, 0.9712],
  "any_lsi": 0.8991
 }
}
```


Message 2: First Look response message format

Definitions of the Response message fields are as follows:

- *segment_id* is the unique string identifier for the segment provided in the input message
- *lsi_predictions* is a dictionary of key-value pairs with the following:
  - Each LSI label (see Table 1) to a vector of confidence scores (value between 0 and 1, inclusive) for presence of LSI in 15 minute time bins starting from *stop_time* in the Predict Message. Vectors must have length equal to *num_*bins provided in the Predict message.
  - *any_lsi* with a single confidence score (value between 0 and 1, inclusive) indicating any LSI occurred within 4 hours of hospital admission

All fields above are required in the Response message. For specification of fields containing embeddings, see Section 3.2.7.

Table 1 contains the string labels for each LSI group expected in the *lsi_predictions* dictionary keys, where LSI groups excluding "Any LSI" correspond to those used in *LSI_table.csv* in the training dataset. Note that LSI groups for Phase 3 have changed from previous phases.

Table 1: LSI group response labels

LSI GROUP	LABEL
Airway – Invasive	`airway_invasive`
Blood Products	`blood_products`
Chest Decompression	`chest_decompression`
Surgical	`surgical`
Vaso/Cardioactive Medications	`vaso_cardioactive_medications`
Any LSI (union of above)	`any_lsi`

After the Response message is received by the Evaluator, the Evaluator will send an Acknowledge message (see Section 3.2.3), and the evaluation will continue with the next segment. A Response message must be received within the segment duration from when the corresponding Predict message was sent, otherwise a Timed Out message is sent by the Evaluator (see Section 3.2.6).

3.2.3 Acknowledge Message Format

The Evaluator will send an Acknowledge message to the Client to indicate successful receipt of a prediction. Here is an example of an Acknowledge message:

```
{
 "case_id": "s1kojt25",
 "segment_id": "OUmJvhCVC5",
 "delta_runtime_sec": 0.2,
 "runtime_remaining_sec": 144000.8,
 "cases_remaining": 240
}
```

Message 3: Acknowledge message format

Definitions of the Acknowledge message fields are as follows:

- *case_id* is the unique string identifier for the current case
- *segment_id* is the unique string identifier for the current segment
- *delta_runtime_sec* is the time elapsed in seconds from Predict message sent to Response message received by the Evaluator, which contributes to the total evaluation runtime
- *runtime_remaining_sec* is the total evaluation runtime limit minus the cumulative runtime in seconds
- *cases_remaining* is the number of cases remaining in the evaluation

There is no specific content required from the Client in response to an Acknowledge message.

3.2.4 Cleanup Message Format

Following the last segment in a case, the Evaluator will send a Cleanup message to the Client indicating the case has ended and the next Predict message will start a new case. Here is an example of a Cleanup message:

```
{
 "case_id": "s1kojt25",
 "runtime_remaining_sec": 144000.8,
 "cases_remaining": 240
}
```

Message 4: Cleanup message format

Definitions of the Cleanup message fields are as follows:

- *case_id* is the unique string identifier for the current case
- *runtime_remaining_sec* is the total evaluation runtime limit minus the cumulative runtime in seconds
- *cases_remaining* is the number of cases remaining in the evaluation

There is no specific content required from the Client in response to a Cleanup message.

3.2.5 Error Message Format

In the event of an error in the evaluation (e.g., malformed Response message from the Client), the Evaluator will send an Error message to the Client. Here is an example of an Error message:

```
{
 “case_id”: “s1kojt25”,
 “segment_id”: “OUmJvhCVC5”,
 “error_message”: <string>
}
```

Message 5: Error message format

Definitions of the Error message fields are as follows:
- *case_id* is the unique string identifier for the current case
- *segment_id* is the unique string identifier for the current segment
- *error_message* is a free-text message describing the error encountered

There is no specific content required from the Client in response to an Error message.

3.2.6 Timed Out Message Format

After a Predict message is sent, the corresponding Response message must be received from the Client before the time elapsed exceeds the segment duration. If no Response message is received before this time limit, a Timed Out message is sent by the Evaluator to the Client.

Here is an example of a Timed Out message sent by the Evaluator to the Client:

```
{
 “case_id”: “s1kojt25”,
 “segment_id”: “OUmJvhCVC5”,
 “delta_runtime_sec”: 0.2,
 “runtime_remaining_sec”: 144000.8,
 “cases_remaining”: 240
}
```

Message 6: Timed Out message format

Definitions of the Timed Out message fields are as follows:
- *case_id* is the unique string identifier for the current case
- *segment_id* is the unique string identifier for the current segment
- *delta_runtime_sec* is the time elapsed in seconds from Predict message sent to Response message received by the Evaluator
- *runtime_remaining_sec* is the total evaluation runtime limit minus the cumulative runtime in seconds
- *cases_remaining* is the number of cases remaining in the evaluation

There is no specific content required from the Client in response to a Timed Out message.

3.2.7 Embeddings Response

To support greater interpretability of and ability to analyze performance of competitors’ models, there are additional fields that can be returned in the Response message. Embeddings response are required from DARPA-funded teams and highly encouraged for self-funded teams.

Many machine learning models operate by constructing vector representations (also called embeddings) of input data before performing additional processing to produce an output prediction. Providing access to this internal data representation can help improve interpretability of model behavior and provide additional surface area to diagnose and resolve issues with models. The schema for returning these embedding values is meant to help support this type of analysis. Given that there are multiple levels at which teams' models may be operating, we suggest fields for embeddings representing both the segment level and the cumulative level, as well as a catch-all "other" category. None of this is prescriptive of how competitors approach the challenge, but rather a best hypothesis for the kinds of model-internal information that may be available.

Here is an example Response message that includes the embeddings field:

```
{
 "segment_id": "OUmJvhCVC5",
 "lsi_predictions": <dict>,
 "embeddings": <dict>
}
```

Message 7: Response message format with embeddings field

Fields are defined as follows and described in detail in subsections below:

- *embeddings* is a dictionary containing embeddings at different levels of internal data analysis

3.2.7.1 embeddings Response Field

The *embeddings* response field contains a dictionary with the following format:

```
{
 "vs_segment": vs_segment_embedding,
 "vs_cumulative": vs_cumulative_embedding,
 "ehr_segment": ehr_segment_embedding,
 "ehr_cumulative": ehr_cumulative_embedding,
 "case_segment": case_segment_embedding,
 "case_cumulative": case_cumulative_embedding,
 "other": [other_embeddings_1,  other_embeddings_2, …]
}
```

Message 8: *embeddings* dictionary format

Note that each individual field in the dictionary above is optional. The value for each embedding vector is expected to be a list of numbers, except for the "other" field, which is expected to be a list of such embedding vectors (i.e., a list of lists).

Embedding vector dimensionality may differ between fields; however, for a given field, the dimensionality should be consistent across all segments. For example, "vs_segment" and "vs_cumulative" embeddings may have different dimensionality, but their respective dimensionality should be the same for each data segment.

Embedding fields description:

`vs_segment`
: An embedding representing the vital signs data from the current data segment.

`vs_cumulative`
: An embedding representing the cumulative vital signs signals across prior segments up to and including this segment.

`ehr_segment`
: An embedding representing the EHR data from the current data segment.

`ehr_cumulative`
: An embedding representing the cumulative EHR data across prior segments up to and including this segment.

`case_segment`
: A higher-level embedding representing the overall case status given the current segment.

`case_cumulative`
: A higher-level embedding representing the overall case status given the cumulative data across prior segments up to and including this segment.

`other`
: Any other embedding(s) that do not fit into the schema above may be included here.

3.2.7.2 Response Message Format, Including Embedding Fields

Given the above specification for the fields "embeddings" and "lsi_predictions", here is an example client response message that includes these fields:

```
{
 "segment_id": "OUmJvhCVC5",
 "lsi_predictions": {
   ...
 },
 "embeddings": {
   "case_segment": [0.152, 0.023, ..., 0.134],
   "case_cumulative": [0.693, 0.193, ..., 0.081]
   "other": [
     [0.232, 0.024, ..., 0.817],
     [0.251, 0.004, ..., 0.948]
   ]
 }
}
```

Message 9: Example response including optional diagnostic fields

3.3 Communication Protocol: Task 2 (Continuous Alert)

For the Continuous Alert task, short consecutive data segments are used to predict LSIs needed in the next 15 minutes. The following subsections describe the contents and format of the following messages sent between the Evaluator and Client: *Predict*, *Acknowledge*, *Timed Out*, *Cleanup*, and *Error*. Where possible, references for equivalent functionality with Task 1 above will be provided.

3.3.1 Predict Message Format

For the Continuous Alert task, the Predict messages contains short (30 second) sequential data segments spanning a single case. The Predict message contains one or more of the following data elements: simulated casualty report, EHR data (including previous interventions), VS data, and metadata. Casualty report and most EHR data will be provided at the beginning of the case, with the remaining data segments primarily containing additional VS data. Here is an example of the Predict message format for a single case:

```
{
 "segment_id": "OUmJvhCVC5",
 "case_id": "wv9ufeR87l",
 "end_of_case": True,
 "start_time": 0,
 "stop_time": 30,
 "casualty_report": <dict>,
 "ehr": <dict>,
 "vs": <dict>
}
```

Message 10: Predict message format for Continuous Alert (Task 2)

Definitions for Predict message fields are as follows:

- *segment_id* is a unique string identifier for the current segment
- *case_id* is a unique string identifier for the current case, composed of many segments
- *end_of_case* is a boolean indicator that this is the last segment for the current case
- *start_time* is the start of the vitals data segment as time elapsed from start of case in seconds
- *stop_time* is the end of the vitals data segment as time elapsed from start of case in seconds
- *casualty_report* is a simulated casualty report similar to the format used for autonomous sensing for medical triage in the Systems Competition (new in Phase 3, details below)
- *ehr* is a dictionary containing Electronic Health Record (EHR) data, described below
- *vs* is a dictionary containing Vital Signs (VS) data, described below

Depending on data availability within each case and segment, the *casualty_report, ehr* and/or *vs* field may be empty dictionaries or *None*. If present, all fields within these dictionaries are optional. For any available data, field names are consistent with the data dictionary and documentation provided with the training dataset.

In response to each Predict message, the Client is expected to respond with a Response message (see Section 3.3.2).

3.3.1.1 ehr *Data Field*

The *ehr* dictionary contains EHR data made available according to the time bounds of the segment as if it was available in real-time during the case. Data are grouped into the following categories:

- **Start-of-Case**. Data available at the beginning of the case, such as GCS taken at the scene, injury type, and general demographic information.

- **At-Admission.** Data available at hospital admission, for example vitals taken at admission.
- **Event Time.** Timestamped data provided according to the segment time window in which they occur, along with timestamps relative to the beginning of the case (in seconds), for example procedures, labs, and medications.

Note that a large portion of the EHR data provided in training datasets will be excluded completely from the evaluation (for example, non-LSI procedures or medications, fields related to outcome or information not available in the acute period of treatment). See Appendix A for list of fields included in the EHR data by run. See the Data Competition Rules Document for more information about evaluation runs.

3.3.1.2 Casualty Report Field

Same as First Look, see Section 3.2.1.2.

3.3.1.3 vs *Data Field*

Same as First Look, see Section 3.2.1.3.

3.3.2 Response Message Format

For Task 2, Clients are expected to respond to the Predict message with predictions for each LSI within a fixed 15-minute time horizon in the future relative to the data segment. Each confidence score must be a value between 0 and 1, inclusive, with no other normalization required across LSIs (i.e., confidence scores need not sum to 1). The *lsi_predictions* dictionary must contain a key for each of the possible LSI labels, excluding *any_lsi* (see Message 11).

Here is an example Response message:

```
{
 “segment_id”: “OUmJvhCVC5”,
 “lsi_predictions”:
 {
  "airway_invasive": 0.2730309253546964,
  "blood_products": 0.06604488945823347,
  "chest_decompression": 0.00642003897813202,
  "surgical": 0.7302976346657081,
  "vaso_cardioactive_medications": 0.28295521562222375
 }
}
```

Message 11: Continuous Alert response message format

Definitions of the Response message fields are similar those provided in Section 3.2.2, with the exception that *lsi_predictions* contain key-value pairs with only LSI labels (no “any_lsi”) and a single confidence score per LSI label. See Table 1 for LSI label list.

All fields above are required in the Response message.

3.3.3 Acknowledge Message Format

Same as Task 1, see Section 3.2.3.

3.3.4 Cleanup Message Format

Same as Task 1, see Section 3.2.4.

3.3.5 Error Message Format

Same as Task 1, see Section 3.2.5

3.3.6 Timed Out Message Format

Same as Task 1, see Section 3.2.6

3.3.7 Embeddings Response

Same as Task 1, see Section 3.2.7.

3.4 Communication Protocol: Task 3 (Resource Allocation)

For the Resource Allocation task, available medical resources are assigned to patients based on predicted need using initial vital signs and health status information. The following subsections describe the contents and format of the following messages sent between the Evaluator and Client: *Predict*, *Acknowledge*, *Timed Out*, *Cleanup*, and *Error*. Where possible, references for equivalent functionality with other tasks above will be provided.

3.4.1 Predict Message Format

For the Resource Allocation task, a series of Predict messages is sent by the Evaluator to the Client for a given scenario: the first Predict message contains resource available within the scenario and subsequent Predict messages contain individual patient data from the patient population. Resource assignments are provided after the last Predict message within the scenario; responses to earlier Predict messages within the same scenario are ignored in the evaluation (see Section 3.4.2).

Each scenario begins with a Predict message containing available medical resources for allocation to patients within the scenario. Here is the initial Predict message format:

```
{
 "case_id": <str>,
 "segment_id": <str>,
 "resources": {
   "hospital_bed": <int>,
   "blood": <int>,
   "ventilator": <int>,
   "surgery": <int>,
   "evacuation": <int>
 }
}
```

Message 12a: Initial Predict message format for Resource Allocation (Task 3)

Definitions for the initial Predict message fields are as follows:

- *case_id* is a unique string identifier for a single scenario within Task 3. Note that unlike Task 1 and 2, this term does not refer to a patient but to a scenario
- *segment_id* is a unique string identifier for this data segment
- *resources* is a dictionary with counts for each resource available within the scenario: *evacuation, hospital_bed, blood, ventilator, surgery*

Subsequent Predict messages within the same scenario (indicated by *case_id*) contain data for individual patients within the scenario. Each Predict message contains the following data types: simulated casualty report, EHR data (previous interventions only), and VS data. Here is the Predict message format containing patient data:

```
{
 "case_id": <str>,
 "segment_id": <str>,
 "patient_id": <str>,
 "casualty_report": <dict>,
 “ehr”: <dict>,
 "vs": <dict>,
 "end_of_case": <bool>,
}
```

Message 12b: Predict message format with patient data for Resource Allocation (Task 3)

Definitions for the Predict message with patient data fields are as follows:

- *case_id* is a unique string identifier for a single scenario within Task 3. Note that unlike Task 1 and 2, this term does not refer to a patient but to a scenario
- *segment_id* is a unique string identifier for this data segment
- *patient_id* is a unique string identifier for a single patient within the scenario.
- *casualty_report* is a simulated casualty report for a single patient similar to the format used for autonomous sensing for medical triage in the Systems Competition (new in Phase 3).
- *ehr* is a dictionary containing Electronic Health Record (EHR) data for a single patient, described below.
- *vs* is a dictionary containing Vital Signs (VS) data for a single patient, described below.
- *end_of_case* is a Boolean indicator for the last patient in the scenario

Depending on data availability for each patient, the *casualty_report, ehr* and/or *vs* field may be empty dictionaries or *None*. If present, all fields within these dictionaries are optional. For any available data, field names are consistent with the data dictionary and documentation provided with the training dataset.

In response to the last patient in the scenario (as indicated by *end_of_case* in the Predict message), the Client is expected to respond with a Response message (see Section 3.4.2). Responses to earlier Predict messages within the same scenario are ignored by the evaluator.

3.4.1.1 ehr Data Field

The *ehr* dictionary contains only EHR data related to interventions received (both within the list of target scored LSIs and other LSIs), with structure similar to First Look (see Section 3.2.1.1). All other patient health data is provided within the Casualty Report field (see next section).

3.4.1.2 Casualty Report Field

Same as First Look, see Section 3.2.1.2.

3.4.1.3 vs Data Field

Same as First Look, see Section 3.2.1.3.

3.4.2 Response Message Format

For Task 3, Clients are expected to respond to the last Predict message within the scenario with assignments of available resources to patients. The Response message contains the list of patients to be evacuated (i.e., to receive any and all LSIs needed without depleting local resources) and the list of patients who are assigned local resources. For each individual medical resource (evacuation included), the number of patients who receive that resource must not exceed the number of available resources in the scenario (provided in the initial Predict message within the scenario). Any assignment of resources exceeding those available within the scenario will result in an error.

Here is the Response message format:

```
{
 "case_id": <str>,
 "evacuated_patients": [<str>, <str>, ...],
 "resource_assignments": [<Assignment object>, <Assignment object>, ...]
}
```

Message 13a: Resource Allocation response message format

```
{
 "patient_id": <str>,
 "resources": [“hospital_bed”, <str>, <str>]
}
```

Message 13b: Assignment object

Definitions of the Response message and Assignment object fields are as follows:

- *case_id* is the unique string identifier for the scenario provided in the input message
- *evacuated_patients* is a list of strings, each a *patient_id* from the list of patients provided in the Predict message
- *resource_assignments* is a list of Assignment objects

- *patient_id* within an Assignment object is a *patient_id* from the list of patients provided in the Predict message
- *resources* within an Assignment object is a list of one or more resources assigned to the patient indicated by *patient_id* within the same object, where each string is a valid label for a local medical resource (see Table 2). This list must contain "hospital_bed", as the baseline medical resources necessary to support any additional specific resources (see section 9.6.4.2 in the Rules document).

All fields above are required in the Response message. The same patient may not appear in both the *evacuated_patients* and the *resource_assignments* lists, and a patient may appear in neither (i.e., not evacuated nor assigned resources).

Table 2 contains the string labels of all possible medical resources expected in the Assignment object list of resources. A patient may be assigned multiple medical resources (e.g., "blood_high" and "airway"). However, a patient may not be assigned multiple instances of the same resource label. For example, a patient cannot be assigned multiple "blood_high" or multiple "airway," resources. Furthermore, a patient may not be assigned both "*blood_high"* and "*blood_low*" resources within the same scenario (see section 9.6.4.4 in the Rules document for more information on resource consumption).

Table 2: Resource response labels

LOCAL RESOURCE	LABEL
Hospital Bed	`hospital_bed`
Blood (High Volume)	`blood_high`
Blood (Low Volume)	`blood_low`
Ventilator	`ventilator`
Surgery	`surgery`

Here is an example response message for the Resource Allocation task:

```
{
 "case_id": "2lknv03jk",
 "evacuated_patients": ["wv9ufeR87l"],
 "resource_assignments": [
  {
   "patient_id": "Bv9LwQ5tYp",
   "resources": ["hospital_bed", "ventilator", "surgery"]
  },
  {
   "patient_id": "ZaX8cDv5fG",
   "resources": ["hospital_bed"]
  },
  {
   "patient_id": "vsj345w7Id",
   "resources": ["hospital_bed", "blood_high"]
```

```
  }

 ]
}
```

Example response message for Resource Allocation

Note that any Response messages sent prior to the last Predict message in the scenario are ignored by the evaluator (see Section 3.4.1).

3.4.3 Acknowledge Message Format

Same as Task 1, see Section 3.2.3.

3.4.4 Cleanup Message Format

Same as Task 1, see Section 3.2.4.

3.4.5 Error Message Format

Same as Task 1, see Section 3.2.5

3.4.6 Timed Out Message Format

Same as Task 1, see Section 3.2.6

3.4.7 Embeddings Response

There is no embedding response for Task 3.

4 Evaluation Environment

Starting with Challenge Event 2, team solutions will be evaluated within AWS. Submissions will be evaluated using instance type **`ml.g4dn.4xlarge`** with the following specifications:

CPU	16 VCPUs
RAM	64 GB
GPU	1
GPU DRIVER	NVIDIA-DRIVER Version 510.47.03
NETWORK	No access
TEST CASES	TBD
TIME LIMITATION	TBD

Test dataset size and time limits for Challenge Event 3 will be provided after Workshop 3 Part 2.

5 Submission Development

Each team must provide code that is compliant with the evaluation process. The following section describes submission requirements, resources, and tools for testing with the JHU/APL evaluation system.

5.1 Requirements

The following subsections describe the minimum requirements for successful submissions.

5.1.1 DTC Base Image

Submissions must be built from an approved DTC Base Image containing the required package *dtc_messaging* for interacting with the Evaluator. The following base images are provided in AWS Elastic Container Registry (ECR):

- *dtc-base-image:latest* for GPU support in Client development
- *dtc-base-image-cpu:latest* for CPU-only support in Client development

Code used to build these images is provided on AWS CodeCommit: https://git-codecommit.us-east-1.amazonaws.com/v1/repos/dtc-base-image

5.1.2 DTC Base Model

Submissions must use a model class that inherits from *DTC_BaseModel* class (included in the *dtc_messaging* package within the DTC Base Image). Callback functions are used to respond to message types described in Section 3. The model class must implement the following callback methods:

- `predict()`: receives a Predict message and returns a Response message
- `acknowledge()`: receives an Acknowledge message, no return message required
- `error()`: receives an Error message, no return message required
- `timed_out()`: receives a Timed Out message, no return message required
- `cleanup()`: receives a Cleanup message, no return message required

An example implementation of the *DTC_BaseModel* class is provided in `template_model.py` in the Client Shell (see Section 5.2).

5.1.3 Docker Entrypoint

The following entrypoint and command will be used to run Client Container submissions within the evaluation system:

```
cmd: "./run_client.py --host <host> --queue <queue> --<run-type>"
entrypoint: ["python3"]
```

where `<host>` and `<queue>` and `<run-type>` will be modified at evaluation time to set the host and RabbitMQ queue name and run type, respectively. The run type for First Look Run 1, First Look Run 2, First Look Run 3, or Continuous Alert tasks will indicated by at most one of the following flags: `--first-look-1`, `--first-look-2`, `--first-look-3`,

`-continuous-alert`, or `-resource-allocation`. Submissions are required to provide `run_client.py` in the working directory of the Docker container. Any additional command line arguments will be ignored during evaluation. Additional parameters required by the submission may be included in a configuration file within the Docker container.

An example implementation of the `run_client.py` script is provided in the Client Shell (see Section 5.2).

5.1.4 Client Log

A volume will be mounted to the Client Container where log files may be saved for delivery back to teams in AWS after an evaluation event. This is an optional feature, and any log files will not impact team scores, nor will they be shared between teams. Any log files should be given unique filenames (e.g., with timestamp) so as not to be overwritten by subsequent evaluation runs. Importantly, EHR or VS data (including derived data) should not be saved in log files.

The following directory will be mounted as a volume to the Client Container for saving log files within the Docker container: /usr/src/app/logs

5.2 Client Shell

To assist in developing compliant submissions, JHU/APL has provided a Client Shell that includes the minimal code needed to create a functioning Client Container. Teams may incorporate their own packages and model-specific code to the Client Shell to build their submission.

The Client Shell contains the following resources:

- `run_client.py`: script used to run the Client
- `template_model.py`: model implementation of *DTC_BaseModel*
- `Dockerfile`: example to build Client Container from *dtc-base-image* and install additional dependencies
- `ReadMe.md`: additional information on Client Shell usage, also available on the AWS wiki at: https://jhuapl-dtc-ta2.github.io/wiki/running_client_shell/

The Client Shell is provided on AWS CodeCommit: https://git-codecommit.us-east-1.amazonaws.com/v1/repos/client-shell

5.3 Evaluator Container Testing

Docker images containing the DTC Evaluator (*dtc-evaluator:latest*) and the RabbitMQ Server (*dtc-rabbitmq:latest*) will be released to participants to assist with testing Clients within AWS prior to submitting for evaluation. See future posts to the DTC forum and documentation on the AWS wiki for more information.

5.4 CodeBuild (CI/CD) Compliance Testing

JHU/APL will provide a Continuous Integration/Continuous Development (CI/CD) system within AWS to give automatic feedback on submission compliance with the evaluation system. This system will containerize selected code using a standard *buildspec* and perform an abbreviated evaluation run to assess the code's compliance with the evaluation system.

To submit code to the CI/CD system, teams should push code to the *compliance-test* branch of their team repository. This action will automatically trigger the CI/CD system to pull the repository and build code with the most recent commit on the *compliance-test* branch. Once the build is complete, the CI/CD system will perform an evaluation run using a small validation dataset to assess compliance with the evaluation system. Any artifacts (e.g., logs) produced by the compliance test will be provided to teams for review within their scratch bucket under the *build_logs* directory.

Usage of the CI/CD system is voluntary, and there is no limit to the number of times a team can test code through the CI/CD system. However, costs related to running the CI/CD system will be subtracted from the team's budget. These costs are expected to be minimal, but code should be pushed to the *compliance-test* branch sparingly to minimize budget usage.

Additional information and release updates to the CI/CD system will be posted to the DTC Forum.

6 Submission Procedure

Code will be pulled down from CodeCommit upon the submission deadline using a specific git tag pushed to the team repository on CodeCommit. Tag nomenclature for event submissions should use the following convention:

submission-phase<PHASE_NUMBER>-<EVENT_TYPE>

where PHASE_NUMBER$\in${1, 2, 3} and EVENT_TYPE$\in${workshop, challenge}. For example, for Workshop 3, submissions should be tagged as:

submission-phase3-workshop

This procedure ensures that a specific, unambiguous commit is evaluated as the official submission. The git tag may be moved to a different commit ahead of the submission deadline, however changing the git tag after submission deadline will not be possible. The submission git tag will also trigger a CI/CD compliance test to ensure submitted code complies with the evaluation system.

7 Appendix A – Data Fields Provided

The following tables provide information about which data fields from UMB and UPitt datasets are provided during evaluation for each task. Columns indicate table name, field name, and when the field appears during evaluation ("Deliver At"). Fields marked "Start" are delivered at the start of the case, and fields marked "Timestamp" are delivered when the event timestamp falls within the data segment start and stop times.

For more details on the dataset contents, see the documentation provided alongside data releases and posted to the DTC Forum.

Tables A.1 and A.2 contain fields provided for LSIs and Casualty Report, respectively. These fields are consistent across UMB and UPitt datasets. For the First Look task, the Casualty Report is only provided in Runs 1 and 2. For the Continuous Alert task, the Casualty Report is provided in the segment 1 for each case. LSI fields are provided for all tasks and runs.

Table A.1: LSI Fields

Table	Field	Deliver At
LSI_table	elapsed_from_start	Timestamp
LSI_table	in_hospital	Timestamp
LSI_table	lsi_description	Timestamp
LSI_table	lsi_group	Timestamp
LSI_table	elapsed_from_adm	Timestamp
other_lsis	elapsed_from_start	Timestamp
other_lsis	in_hospital	Timestamp
other_lsis	lsi_description	Timestamp
other_lsis	lsi_group	Timestamp
other_lsis	elapsed_from_adm	Timestamp

Table A.2: Casualty Report Fields

Table	Field	Deliver At
casualty_report	sex	Start
casualty_report	age_years	Start
casualty_report	weight_kg	Start
casualty_report	height_cm	Start
casualty_report	hr	Start
casualty_report	rr	Start
casualty_report	severe_hemorrhage	Start
casualty_report	respiratory_distress	Start
casualty_report	trauma_head	Start
casualty_report	trauma_torso_front	Start

casualty_report	trauma_torso_back	Start
casualty_report	trauma_arm_left	Start
casualty_report	trauma_leg_left	Start
casualty_report	trauma_arm_right	Start
casualty_report	trauma_leg_right	Start
casualty_report	trauma_arm_unspecified	Start
casualty_report	trauma_leg_unspecified	Start
casualty_report	trauma_unspecified	Start
casualty_report	alertness_ocular	Start
casualty_report	alertness_verbal	Start
casualty_report	alertness_motor	Start
casualty_report	description	Start

Tables A.3 and A.4 contain the EHR fields for UMB and UPitt cases, respectively. For the First Look task, additional EHR data is only provided in Run 3. For the Continuous Alert task, EHR data is provided within the segment that it occurs ("Timestamp" data provided according to time within data segment, "Start" and "Admission" data is provided in segment 2 after the Casualty Report).

Table A.3: UMB-specific EHR Fields

Table	Field	Deliver at
demo_scores	AGE	Start
demo_scores	HEIGHT	Start
demo_scores	INJTYPEID	Start
demo_scores	INJURYTYPEDESCRIP	Start
demo_scores	RTS_S	Start
demo_scores	SEXID	Start
demo_scores	WEIGHT	Start
demo_scores	ADM_DYSBP	Admission
demo_scores	ADM_GCS_EYE	Admission
demo_scores	ADM_GCS_MOTOR	Admission
demo_scores	ADM_GCS_VERBAL	Admission
demo_scores	ADM_HR	Admission
demo_scores	ADM_O2SAT	Admission
demo_scores	ADM_RR	Admission
demo_scores	ADM_SYSBP	Admission
demo_scores	ADM_TEMP	Admission
demo_scores	GCSTOTAL	Admission
demo_scores	RTS_A	Admission
pta_vitals	PTA_DBP	Start

pta_vitals	PTA_GCS_E	Start
pta_vitals	PTA_GCS_M	Start
pta_vitals	PTA_GCS_TOTAL	Start
pta_vitals	PTA_GCS_V	Start
pta_vitals	PTA_HR	Start
pta_vitals	PTA_RR	Start
pta_vitals	PTA_SBP	Start
pta_vitals	PTA_TEMP	Start
pta_vitals	elapsed_from_adm	Start
pupillometry	%L	Timestamp
pupillometry	%R	Timestamp
pupillometry	CVL	Timestamp
pupillometry	CVR	Timestamp
pupillometry	date	Timestamp
pupillometry	DVL	Timestamp
pupillometry	DVR	Timestamp
pupillometry	elapsed_from_start	Timestamp
pupillometry	LatL	Timestamp
pupillometry	LatR	Timestamp
pupillometry	MCVL	Timestamp
pupillometry	MCVR	Timestamp
pupillometry	MinL	Timestamp
pupillometry	MinR	Timestamp
pupillometry	NPiL	Timestamp
pupillometry	NPiR	Timestamp
pupillometry	SizeL	Timestamp
pupillometry	SizeR	Timestamp
pupillometry	elapsed_from_adm	Timestamp
ems	LANDVSAIR	Start

Table A.4: UPitt-specific EHR Fields

Table	Field	Deliver At
events	BloodPressureDiastolic	Timestamp
events	BloodPressureSystolic	Timestamp
events	BPMethod	Timestamp
events	Carboxyhemoglobin	Timestamp
events	ECGMethod	Timestamp
events	elapsed_from_start	Timestamp

events	EndotrachealCO2	Timestamp
events	EndotrachealCO2Type	Timestamp
events	GCSEye	Timestamp
events	GCSMotor	Timestamp
events	GCSVerbal	Timestamp
events	Glucose	Timestamp
events	HeartRate	Timestamp
events	HeartRateMethod	Timestamp
events	LevelofConsciousness	Timestamp
events	MeanArterialPressure	Timestamp
events	OxygenSaturation	Timestamp
events	Procedure	Timestamp
events	PulseRhythm	Timestamp
events	Respiration	Timestamp
events	RespiratoryEffort	Timestamp
events	RhythmCoded	Timestamp
events	TempDegreesType	Timestamp
events	Temperature	Timestamp
events	TemperatureObtained-Method	Timestamp
events	VitalRhythms	Timestamp
events	elapsed_from_adm	Timestamp
injurydetails	InjuryCause	Start
injurydetails	InjuryMechanism	Start
injurydetails	InjuryMechanism1	Start
injurydetails	InjuryMechanism2	Start
injurydetails	InjuryMechanism3	Start
neuro	ChemParalyzed	Start
neuro	InitialGCSEye	Start
neuro	InitialGCSMotor	Start
neuro	InitialGCSTotal	Start
neuro	InitialGCSVerbal	Start
neuro	RevisedTraumaScore	Start
neuro	RevisedTraumaScoreBP	Start
neuro	RevisedTraumaScoreResp	Start
patient	AgeInYears	Start
patient	AgeType	Start
patient	BloodType	Start
patient	Gender	Start

patient	Height	Start
patient	HeightType	Start
patient	PatientAge	Start
patient	Weight	Start
patient	WeightType	Start
proclabs	elapsed_from_start	Timestamp
proclabs	LabLactateArterial	Timestamp
proclabs	LabLactateVenous	Timestamp
proclabs	LabsHCO3	Timestamp
proclabs	LabsPCO2	Timestamp
proclabs	LabspH	Timestamp
proclabs	LabspO2	Timestamp
proclabs	elapsed_from_adm	Timestamp
general	InterhospitalScene	Start