**Logging System Structured Documentation**

**First Layer – Layer of General Information**

The first layer of the logging system contains information that is general for all kinds of event types. Based on the logging system of MHS1.0, the variables for the general information include:

* **itemID:** A unique Id given to each collected log record
* **installID:** A unique Id given to each computer (or device), which installs the game.
* **buildType:** What type is the game release version, “Development” for within-team testing, or “Production” for field tests in schools.
* **buildVersion:** A unique Id given to each released game version, which we can base on to select which game version we should analyze with.
* **playerName:** Name of the student (player), can be a name given by the teacher and the teacher has the name map of the students’ fake names and real names.
* **playerId:** a unique ID for a student, in case the playerName may have duplicated records.
* groupType: Whether the player is leaded by an instructor within a class setting or plays the game individually.
* **platform:** which platform the student used to play the game (e.g., Windows, mac, iPad, as such).
* **timestamp:** The time point when this log record is collected with a format of “yyyy-mm-dd hh:mm:ss”.
* **timezone:**  Time zone (should be meaningful): still need more discussion (restriction of geographic zones where the game will be played).
* **playerPosition:**
  + X
  + Y
  + Z
* **cameraRotation:**
  + X
  + Y
  + Z
* **performance:**
  + memory
  + frame/sec
  + others (to be continued)
* **replayNumber:** The number of how many times the student has replayed corresponding game parts.
* **sceneNames**: Which scene the student (player) is in (each unit has several scenes). Example name format: “Unit 1\_space ship” “Unit 2\_ailien dungeon” “Unit 2\_topotraphy.”
* **regionName:** The region name within a certain scene name.
* **questTable**: Which quest the student has in their quest menu.
  + **activeQuests:** An array of quests that are currently active.
  + **completedQuests:**An array of quests that have been completed.
* **taskTable**: Which task the student has in their task menu.
  + **activeTasks:** An array of tasks that are currently active.
  + **completedTasks:** An array of tasks that have been completed.
* **eventType:** Which event type this record belongs to. Example events include: gameStartEvent, gameQuitEvent, triggerEvent, questEvent, taskEvent, dialogueEvent, dialogueNodeEvent, movementEvent, DANIMenuEvent, hotkeyEvent, DANIFeatureEvent, argumentationEvent, argumentationNodeEvent, argumentationAnswerEvent, argumentationToolEvent, miniGameEvent, miniGame InteractionEvent, topographicMapEvent, objectivesEvent.
* **specificEventDetail:** This variable is structured as a nested JSON format and contains multiple variables belonging to a specific event type. Not all event types have value for this variable.

**Second Layer – Layer of Specific Information for Each Event Type**

**Game Start Event**

No event type specific detailed information.

**Game Quit Event**

No event type specific detailed information.

**Trigger Event (Object Interaction Event)**

* **objectID:** A unique id created for each object in a certain scene, so that we can identify which object the player (student) interacts with.
* **objectName:** A name of the object, such as “Cube1inU2Dungeon,” “ControlPanel1inU3Dungeon.”
* **actionType:** The name of the action, such as “Lift,” “Drop,” “Enter,” “Select,” “Press.”
* **objectStatement:** After solving some puzzles, some in-game objects will change their statement (e.g., colors, format, or outcome). If no statement changing, we can put a value like “NA” or “Not Applicable.”

**Quest Event**

* **questID:** A unique Id given to each quest.
* **questName:** The description name given to a quest Id, example names include “U1 – Welcome to WAT247,” “U1 – An Amazing AI,” and “U1 – Time to Argue.”
* **questEventType:** Two event types under this variable – “QuestActiveEvent,” and “QuestCompleteEvent.” “questActiveEvent” represents the log record when the student accepted a new quest. And “questCompleteEvent” represents the log record when the student completed a quest.

**Task Event**

* **questID:** A unique Id given to each quest.
* **taskID:** A unique Id given to each task.
* **taskName:** The description name given to a task, example names include “wakeup,” “arfRoom,” “topoTalk.”
* **taskEventType:** Three event types under this variable – “taskActiveEvent,” “taskCompleteEvent,” and “taskProgressEvent.” “taskActiveEvent” represents the log record when the student accepted a new task. And “taskCompleteEvent” represents the log record when the student completed a task. In terms of “taskProgressEvent,” some tasks can be broken down into several parts, each part has an achievement point. This task event type is to mark whether the player get the achievement. For example, there is a task in unit 2 of MHS2.0 that needs the player to collect evidence around the waterfall by flying a drone. After successfully collecting a piece of evidence, the corresponding piece of evidence will be checked on the evidence panel. This task event type is to record such achievements.
* **progressContent:** This variable is to save what in-task progress the player has achieved, such as what evidence has been collected.

**Dialogue Event**

* **dialogueID:** A unique ID given to each dialogue section.
* **dialogueEventType:** There are two values under this variable: “DialogueStartEvent” and “DialogueFinishEvent.” “DialogueStartEvent” will be generated when players start a dialogue section, while “DialogueFinishEvent” will be generated when players end a dialogue section.”

**Dialogue Node Event**

* **dialogueID:** A unique ID given to each dialogue section.
* **dialogueNodeID:** A unique ID given to the dialogue box, which contains the choice nodes, within the corresponding dialogue section.
* **choiceID:** The ID of the choice, which is selected by the student. (Leave it as Null vale when the dialogueNodeEventType equals to “DialogueNodeStartEvent.”)
* **dialogueNodeEventType:** There are two values under this variable: “DialogueNodeFinishEvent” and “DialogueNodeStartEvent.” “DialogueNodeStartEvent” will be generated when players start a dialogue box that contains choice nodes, while “DialogueNodeFinishEvent” will be generated when players made a choice and move on to the next dialogue box.

**Movement Event**

* **actionName:** What specific action name of this movement log record, potential value of this variable could be “forwardMove,” “leftMove,” “rightMove,” “backMove,” “jump,” or corresponding keyboard – “W,” “A,” “D,” “S,” and “Space.”
* **state:** Whether it’s a beginning or ending of a movement events, example values could be “start,” and “end.”
* **isToggleHoverboardUsed:** This is a binary variable containing “Yes” and “No.” If it is “Yes,” then the player is using Toggle hoverboard to navigate within the game world, otherwise they are not using the hoverboard to navigate within the game.

**DANI Menu Event (Use “Tab” button to open the panel with in-game tools)**

* **actionName:** There are two actions within this event type – “DANIMenuOpenEvent” and “DANIMenuCloseEvent”. “DANIMenuOpenEvent” happens when the player presses the “Tab” keyboard button to open the in-game tool panel. “DANIMenuCloseEvent” happens when the players closes the in-game tool panel.

**DANI Feature Event (Choose a feature within the in-game tool panel)**

* **featureName:** The name of the tool. Example names include “map,” “background info,” and “chat log.”
* **actionName:** There are two actions within this event type – “arfFeatureOpenEvent” and “arfFeatureCloseEvent”. “arfFeatureOpenEvent” happens when the player presses the “Tab” keyboard button to open the in-game tool panel. “arfFeatureCloseEvent” happens when the players close the in-game tool panel.

**Hotkey Event**

* **keyName:** Which keyboard button the player pressed. Example keyboard buttons include “Tab,” “B,” “M,” and “Q.”
* **description:** A description name given to each keyboard button. For example, “Tap” equals to arf, “B” equals to backingInfo, “M” equals to map, and “Q” equals to quests.

**Topographic Map Event**

* **featureUsed:** Which feature on the map is used, including topographic level button and waypoint.
* **actionType:** What specific action the player conducts. Such as “Drag,” “Drop,” “Select,” and “Unselect.”
* **locationInfo:** Where the player moved the waypoint to on the topographic map.

**Objectives Event**

* **action:** What action is the player performing within the “objectives” menu, including “select” and “unselect”.
* **missionSelect:** Which mission the player chose to track. Such as “Foraged Forging: Collect 5 piles of scrap metal.”

**Argumentation Event**

* **argumentationTitle:** The title or topic of the argumentation. Example names include “U1 – Argumentation tutorial,” “U2 – Watershed,” and “U3 – Pollution Upstream.”
* **argumentationDescription:** A brief description of the corresponding argumentation title. Such as “U1 – Argumentation tutorial” equals to “Place the claim, reasoning, and evidence orbs in orbit,” “U2 – Watershed” equals to “Which watershed is bigger based on collected evidence from eastern and western waterfalls,” and “U3 – Pollution Upstream” equals to “Where is the pollution site probably located?”
* **arctionType:** There are two action types under this event type – “argumentationSessionOpen” and “argumentationSessionClose.”“argumentationSessionOpen” happens when the player opens the scientific argumentation engine, while “argumentationSessionClose” happens when the player closes the scientific argumentation engine.

**Argumentation Node Event**

* **argumentationTitle:** The title or topic of the argumentation. Example names include “U1 – Argumentation tutorial,” “U2 – Watershed,” and “U3 – Pollution Upstream.”
* **nodeName:** The name of the choice node. Example names include “A,” “1,” and “II.”
* **arctionType:** There are four action types under this event type – “argumentationNodeHoverStart,” “argumentationNodeHoverEnd,” “argumentationNodeAdd,”and“argumentationNodeRemove.” “argumentationNodeHoverStart” happens when the player hovers on a choice node and a dialogue box will pop up for them to read. “argumentationNodeHoverEnd” happens when the player ends hovering on a specific node. “argumentationNodeAdd” happens when the player drags the choice node into the argumentation structure, while “argumentationNodeRemove” happens when the players removes the node from the argumentation structure.

**Argumentation Answer Event**

* **argumentationTitle:** The title or topic of the argumentation. Example names include “U1 – Argumentation tutorial,” “U2 – Watershed,” and “U3 – Pollution Upstream.”
* **answerSubmitted:** A combination of the three components – Claim, Reasoning and Evidence. Examples include “I, A, 2,” “II, B, 1,” and “I, C, 7.” If only one component needs to be selected, then it will be – “I,,” if only Claim should be selected, “,A,” if only Reasoning should be selected, and “,,1” if only evidence should be selected.
* **arctionType:** There are two action types under this event type – “argumentationSuccess,” and “argumentationFailed.” “argumentationSuccess” happens when the player hit the “Submit” button with the correct answer, while “argumentationFailed” happens when the player submitted with a wrong answer.
* **feedbackText:** What feedback the player received when submitting the answer. Feedback texts depend on what answer the player submitted.

**Argumentation Tool Event**

* **argumentationTitle:** The title or topic of the argumentation. Example names include “U1 – Argumentation tutorial,” “U2 – Watershed,” and “U3 – Pollution Upstream.”
* **toolName:** The name of the tool that the player can refer to within the argumentation engine.
* **arctionType:** There are two action types under this event type – “argumentationToolOpen,” and “argumentationToolClose.” “argumentationToolOpen”represents the player opens the argumentation tool within a certain argumentation session, while “argumentationToolClose” represents the player closes the argumentation tool within a certain argumentation session.

**~~(CREi System Event~~**

* **~~itemID:~~** ~~A unique Id given to each collected log record, same as that in the first layer of the logging system, so that we can merge two tables together.~~
* **~~actionType:~~** ~~There are two action types within this event type, which are “ballSpawn,” “ballDelivery,” “CREiSystemOpen,” “CREiSystemClose.” “ballSpawn” represents the player who receives a ball and spends time to read the question and requests. And, “ballDelivery” represents the player who makes the decision and delivers the soccer ball in a certain direction. “CREiSystemOpen” happens when the player enters into the CREi system, while “CREiSystemClose” happens when the player leaves the system.)~~

**~~(CREi Answer Event~~**

* **~~itemID:~~** ~~A unique Id given to each collected log record, same as that in the first layer of the logging system, so that we can merge two tables together.~~
* **~~CREiArgQuestion:~~** ~~The argumentation title of this question session. Example questions include “What is ice,” “Does the East well have cleaner water than the west well,” and “Is the water polluted?”~~
* **~~CREiAnswerCompType:~~** ~~The answer component decided by the player. Such as Claim, Reasoning or Evidence.~~
* **~~CREiAnswerCompDesc:~~** ~~The description the player based on to make the component choice.~~
* **~~CREiRealAnswer:~~** ~~The correct answer of the corresponding description. Such as Claim, Reasoning or Evidence.~~
* **~~CREiAnswerCorrct:~~** ~~Whether the player selected the correct answer. “True” if the answer is correct, “false” if the answer is wrong.~~
* **~~Feedback:~~** ~~The feedback (usually within a dialogue box) the player received after delivering a ball.)~~

**Mini Game Event**

* **gameSession:** Which mini game session does this log belongs to, such as “Nanomanchine1,” “Nanomanchine2,” and “Forge Mini Game.”
* **action:** it shows whether the player just starts the session or completes the session. Potential value could be “start” and “end.”

**Mini Game Interaction Event**

* **gameSession:** Which mini game session does this log belongs to.
* **action:** What action the player is playing, such as “drag,” “drop,” and “remove”
* **object:** Which piece is interacted by the player, such as “horizontalLine1,” “horizontalLine2,” “leftDownCorner,” and “rightDownCorner.”
* **statement:** How the action on the object change the statement of the object, or the outcome of using the object, such as “cold,” “warm,” or “hot.” Not all interactions have such value, if not, just put an NA there.

**Third Layer – Layer of Reference Information for Each Event Type (Can be Saved in Local instead of being collected by the logging system)**

**Dialogue Event**

* **dialogueID:** A unique ID given to each dialogue section.
* **dialogueNames:** A description name given to each dialogue section.
* **rawTextContent:** This variable contains all texts that will be shown under a specific dialogue section. (Notably, each dialogue section may contain multiple dialogue boxes. I combined all dialogue boxes’ content into this variable.)

**Dialogue Node Event**

* **dialogueID:** A unique ID given to each dialogue section.
* **dialogueNodeID:** A unique ID given to the dialogue box, which contains the choice nodes, within the corresponding dialogue section.
* **choiceID:** Available choice IDs in the corresponding dialogue node.
* **dialogueNodeName:** A description name given to the dialogue box where the choice nodes are contained.
* **dialogueNodeText:** The text content shown in a specific dialogue box, containing choice nodes.
* **choiceText:** The corresponding text content under each choice ID.

**Argumentation Node Event**

* **argumentationTitle:** The title or topic of the argumentation. Example names include “U1 – Argumentation tutorial,” “U2 – Watershed,” and “U3 – Pollution Upstream.”
* **nodeName:** The name of the choice node. Example names include “A,” “1,” and “II.”
* **nodeDescription:** The full text of the corresponding choice node. Such as “Water that evaporates from the ocean will leave the salt behind. Thus the rain that falls on land will be drinkable.”

**Player General Information**

* **playerId:** a unique ID for a student, in case the playerName may have duplicated records.
* **teacherId:** Which teacher leads the class
* **classId:** Which class the student belongs to