EXODE QUICK SHEET

API COMMANDS:
EXODE MISSIONS VIEW

Version M2a
July 3rd 2024

How to set up a third party program

OVERVIEW

Setting up your third party program is as simple **as sending one API call**, which will get **a full JSON package** with all possible mission information. In all the mission data:

- The "missionContext" array has all information on the mission location,

- The "teamSlots" array has all information on participating Agents,

- The "animationLog" array has all actions done during the mission itself: be it when an event is triggered, when someone participated, what rolls were performed, etc.

These three tables should provide you with all important dynamic information, although other fields in the JSON array can be found of interest to you. Additionally, you can be interested in the "objectives" which are targets for the mission, "detailedLog", which has more human-readable data and logs, or in the "currentEvents" table for information on the events that the mission received.

Your base process should be to read the animationLog and animate things onscreen according to what you read.

(1) MAKING YOUR API CALL

We use a small and very simple sample of our own development framework, called BAYONET, to deliver your API content.

We have our script example in our own mission viewer (https://exode.io/game/mission-report/js/bayonet_network_tools_demo.js). Please be reminded that we do not focus on programming structure here.

First you need to set up two global variables:
player_accountName with the account id (ie. elindos)
player_missionId with the mission id (it is a MD5 string)

Then you can use it with:

```
bayonet_apiAction("ecgPublic", "loadMissionContent",
{"missionId":player_missionId,"account":player_accountName},
function(_feedbackContentOk) {

	// your code here in case your data is OK
	// …


}, function(_feedbackError = "Unknown error") {

	// your code here in case there is an error
	// …


});
```

You have an example of this currently in our start script.
(https://exode.io/game/mission-report/js/start_script.js)

You are authorized to include or copy the network script in your build, and to use the code above.

(2) INTERPRETING RECEIVED DATA

“missionContext”

Mission context is an array which you are not forced to use but which has relevant procedural data created when the user received the mission in the first place.

It is also, actually, actively used during events and during the last steps when the colony leader performs his “Mission Review”.

Almost all of this data is actually “persisting”: your mission location was generated with its attributes, its colony leader, its planetary type and dangers. Some attributes can be modified or used during the mission itself.

The missionContext will evolve during later versions. It currently has one immense attribute called “location” with everything else inside it.

The location has:

an **id** (a MD5 string),
a **name** array (with any name array, you can use [“name”][“firstName”] or [“name”][“lastName”] for the full name, or directly [“name”][“name”])
and an **attributes** array.

Attributes are where things get interesting.

You will really want to read the **leader_data** information, and his **name**, **tastes**, and of course face picture (**leader_data[“faceUrl”]**).

We often use https://jsonviewer.stack.hu/ for presenting a JSON package, but you can use anything you fancy as there are many JSON beautifiers available on the internet.

(3) INTERPRETING RECEIVED DATA

“teamSlots”

The teamSlots array has one line for every opened agent slot on a mission ; if there is an agent assigned, then the corresponding line will have some other data such as:

Property	Example value	Description
asset_role	colonyConstructionBuilder	This is the code of the role the player has decided to assign to this Agent. It is one of the codes allowed for this slot (“possibleRoles”).
asset_globalid	exode_card_004_officerWeapons	A “globalid” is an EXODE identifier of a card template id. This is normally what represents most of your Agent base abilities and upgrades options. There are hundreds of card templates in EXODE, and they usually are represented with exode_card_XXX for EXODE cards, for instance.
asset_character	Array	This one has a lot of attributes, some of which could be removed, taken from the global template and added from instant character generation. The mission system generated an EXODE character with fixed average rolls to determine skill values. The mission system then made use of the character skill values and traits when participating in events and making rolls. So the fact the mission system also copies the generated Agent data used for these rolls is just some form of technical curtesy.
asset_nbDice	3	This is the number of dice the Agent will use during the mission
asset_nbDice_average	3	Not used at the moment. It is the number of dice, but determined from skill values with another system.
missionEnergy	100	These values should be the attribute that the Agent started with in the mission.
…		…

You should use the “picture_url” of the asset_character and other information to represent the Agent during the mission.

(4) INTERPRETING RECEIVED DATA

“animationLog”

This is definitely the most important data for animating your mission view.

The animationLog has every single action or event relevant to mission execution, an index of who did it and what rolls were made.

Note : this data will evolve A LOT during later iterations of missions. You have been warned!

Code	Other data	Description
startMajorEvent	event index in currentEvents(ie. 0)	Means that an EVENT is triggered. Once an EVENT is launched your sequence should switch to EVENT RESOLUTION actions, detailed further below, until you receive an “endMajorEvent”. Usually during an event, Agents face a challenge, will take decisions, make rolls to succeed and decisions will have effects on the mission.
endMajorEvent	event index in currentEvents	Means that current EVENT RESOLUTION should end and you can now switch to normal mission resolution.
focusOnTeamSlot	slot index in teamSlots (ie. 0)	This one has no effect but just to warn you that an Agent is going to take the next action. In our official mission viewer, we use this to reduce opacity of other agents and put some focus on the main one.
makeRolls	slot index in teamSlots (ie. 0) rolls made (array) with value, and is it a success (true or false), and total successes value	The character at this index is rolling a number of dice. This usually means that he is testing one of his MISSION ROLE abilities.
actionMessage	slot index in teamSlots (ie. 0) message (string)	This announces what the character is doing, in clear text.
nextRound		Means we get to current round + 1. The mission has rounds and some objectives must be completed within a set timer.
(effects)	slot index in teamSlots (ie. 0) or objective index in objectives	There are many codes which can be used for mission effects. Usually they are spread between buffs and progression on objectives. You usually have an actionMessage next which explains what happened ; unless it is “progressionOnTask” which means we progressed on an objective
…	…	…

completedTask	objective index in objectives	Means that one of the objectives is done for, so we get to the next one, if any.
completedMission		Means the current mission is a SUCCESS. In case of colony support, this also is normally followed by the MISSION REVIEW by colony leader.
failedMission		Normally means that the current mission is a FAILURE. In case of colony support, this also is normally followed by the MISSION REVIEW by colony leader.
reviewingTaste	taste index in "tastes" of leader_data taste code (ie. faction, criticalQuality, fastSpeed, etc)	Means that the colony leader is reviewing the mission according to one of his tastes.
completedMissionGainReputation	positive or negative value	This means that according to review, player reputation is directly affected by the value.
…	…	…

EVENT RESOLUTION

Code	Other data	Description
agentPicksDecisionCode	slot index in teamSlots (ie. 0) decision code from currentEvents available decisions	Means that according to his skills and traits, , an Agent decided to participate in the EVENT by taking the mentioned decision. If several decisions are possible for an Agent (he/she has an equal preference for any) then one decision among them was already randomly picked.
agentPicksDecisionAtIndex	slot index in teamSlots (ie. 0) decision index from currentEvents available decisions	Duplicate of above, but uses decision index instead of decision code, so maybe it can be more practical to you.
agentDecisionSuccessChances	slot index in teamSlots (ie. 0) chances value (in percentage)	This means that the Agent will have the mentioned chances to succeed in the decision he took.
(decision outcome)	slot index in teamSlots (ie. 0) roll which was made (with a d100)	The results of a decision is usually one of: "agentDecisionIsCriticalSuccess" "agentDecisionIsSuccess" "agentDecisionIsAutoSuccess" "agentDecisionIsFailed"
agentDecisionTextMessage	slot index in teamSlots (ie. 0) message (string)	This announces what the character has been doing when succeeding or failing at the decision outcome.
(decision effects)		This can be numerous lines in animationLog, affecting chances for the EVENT to be a success or not, or giving buffs or debuffs to the team.

During an Event Resolution, Agents make decisions, roll for the results, and this can effect the global outcome of the Event itself or give mission lasting buffs and debuffs, reputation effects, etc.

The event ends with an “eventEndMessage” mentioning with a string what the outcome was

After this line, a number of event effects occur, until you get the “endMajorEvent” line.
After the “endMajorEvent”, you should get back to normal mission resolution.

At the moment, missions can receive a starting event, and then normal mission resolution begins, but things can be different in other missions.

Thanks and good luck!

This ends this Quick Sheet for the moment!

Thank you for participating in EXODE Development.

If you have any questions, you can of course ask them out, but please note that we will certainly be busy working on a new EXODE scene or feature. Do not hesitate, however to make the community know what you are working on!

Rewards for third parties will also be revealed later: we plan to distribute API keys to developers, watch how intensive their builds are used by players, and reward them according to usage by all these accounts.