Specification: Primary Game Data File

Purpose

This JavaScript file serves as the primary data store for a client-server strategy web application. It provides a centralized source of game state and configuration information, which is:

1. **Read and written on the server** via PHP using `json_decode()` and `json_encode()`.
2. **Read directly by the client** in JavaScript to render UI elements and execute game logic.

The file contains structured JSON-like objects representing game entities such as colonies, fleets, events, units, maps, and player/empires. It is intended to be authoritative for game state.

Data Structures

1. Colonies

Variable: `colonies`
Type: Array of objects
Each colony object contains the following properties:

Property	Type	Description
`name`	string	Name of the colony.
`capacity`	integer	Maximum statistics of the colony
`fort`	integer	Level of fortification.
`intel`	integer	Intelligence coverage.
`morale`	integer	Morale of the colony population.
`owner`	string	Owning empire, empty if unowned.
`population`	integer	Current population of the colony.
`raw`	integer	Resource richness index.
`type`	string	Environmental type (e.g., "Barren", "Adaptable", "Homeworld", "Dead", "Extreme", "Garden").
`notes`	string	Special features or ongoing events in the colony. Includes “Martial Law”, “Blockaded”, “Rebellion”, and “Opposition”. See also section 2.1.5
`fixed`	array	Array of fixed structures or units that cannot move on their own, at the colony.

2. Empire

Variable: `empire`
Type: Object
Represents the player's empire.

Property	Type	Description
`empire`	string	Empire identifier.
`maintExpense`	integer	Maintenance expense.
`miscExpense`	integer	Miscellaneous expenses.
`miscIncome`	integer	Miscellaneous income.
`name`	string	Full name of the empire.
`systemIncome`	integer	Income from all controlled systems.
`previousEP`	integer	Economic points carried over from previous turn.
`techYear`	integer	Current technological year.
`tradeIncome`	integer	Income from trade.
`researchInvested`	integer	Resources invested in research.

3. Events

Variable: `events`

Type: Array of objects

Events capture abnormal happenings to an empire. This is not intended as a checklist of what has been done or what needs to be done. Rather, it collects things that need to be brought to the attention of the player. As a non-exhaustive list, this should contain items such as successful or unsuccessful raids, a change in diplomatic standing with another empire, fleets being encountered, battles to be resolved, and covert operations by or to the player. Each event object contains:

Property	Type	Description
`event`	string	Short event description.
`time`	string	Turn of event.
`text`	string	Detailed narrative or calculation for the event.

4. Fleets

Variable: `fleets`

Type: Array of objects

Each fleet object contains:

Property	Type	Description
`name`	string	Fleet identifier.
`location`	string	Current colony or sector.
`units`	array	List of ship/unit names in the fleet.
`notes`	string	Optional notes regarding fleet.

5. Game State

Variable: `game`

Type: Object

Property	Type	Description
`game`	string	Name of the game.
`turn`	integer	Current turn number.

Property	Type	Description
`monthsPerYear`	integer	Game months per year.
`blankOrders`	integer	Number of blank orders available.
`turnSegment`	string	Current phase of turn (e.g., "pre", “post”).
`nextDoc`	string	Identifier of next saved game file.
`previousDoc`	string	Identifier of previous saved game file.
`techAdvancement`	Boolean	If true, then use the Historical Tech Advancement optional rule(5.18). Else, use research as written.

6. Map Data

Map Points: Array of coordinates representing colonies on the map.

Variable: `mapPoints`

Type: Array

Type	Description
integer	Y coordinate to place the map space. These are 35 units apart
integer	X coordinate to place the map space. These are 60 units apart
string	Owner of the map space. Same string as is found at empire[‘empire’]
string	Name of the system at that map space

Map Connections: Array representing known or unexplored routes between colonies.

Variable: `mapConnections`

Type: Array

Type	Description
string	System name to source the connection. Same string as is in mapPoints[3]
string	System name to end the connection. Same string as is in mapPoints[3]
string	Status of the connection. Can be “Unexplored”, “Restricted”, “Minor”, “Major”

7. Orders, Treaties, and Projects

Orders: List of player-issued actions in the format of the following:

Variable: `orders`

Type: Array

Property	Type	Description
`type`	string	Order Type
`receiver`	string	Thing being ordered
`target`	string	Where to perform order
`note`	string	User-entered value
`perm`	boolean	If 0, show order in drop-downs if 1, no drop down. Show text

Offered Treaties: Array of treaties offered by other empires.

Variable: `offeredTreaties`

Type: Array

Type	Description
string	Empire that offered the treaty. Same string as is found at empire[‘empire’]
string	Type of treaty. Can be “War”, “Hostilities”, ”Neutral”, “Non-Aggression”, “Trade”, “Mutual Defense”, “Alliance”

Treaties: List of treaties with other empires

Variable: `treaties`

Type: Object

Property	Type	Description
`cooldown`	int	Diplomatic cooldown value
`empire`	String	The empire this affects
`type`	string	Political state: War, Hostilities, Neutral, Non-Aggression, Trade, Mutual Defense, Alliance
`income`	int	Known income of other power
`navy`	int	Known naval construction value of other power

Intel Projects: List of ongoing espionage or sabotage projects

Variable: `intelProjects`

Type: Object

Property	Type	Description
`type`	string	Type of mission
`target`	string	What system the mission is affecting
`location`	string	Where the mission is sourced

8. Units and Purchases

Unit List: Contains general unit information of all units that might be built by this empire.

Variable: `unitList`

Type: Object

Property	Type	Description
`ship`	string	Ship designator or class (e.g. “FFE”, “BCH”)
`yis`	int	The year that the ship becomes available
`design`	string	The ship hull designation (e.g. “CA”, “AB”)
`cost`	int	The cost of the unit, in EPs
`notes`	string	Any notes on the units of this class. Comma deliminated list. Some traits are followed by a number in parenthesis. See also section 2.5
`researched`	boolean	Tracks if the unit is available for construction via the research mechanism. Use for all cases of ability to construct

Purchases: List of units that are currently being purchased by the empire.

Variable: `purchases`

Type: Object

Property	Type	Description
`cost`	integer	Amount spent on this turn to create this item
`name`	string	The name of the item being created.

Under Construction: Tracks units currently being built at specific colonies.

Variable: `underConstruction`

Type: Object

Property	Type	Description
`location`	string	Place that the unit is being built
`unit`	string	The unit being built. See `Purchases` for cost.

Units in Mothballs: Same format as fleets, but always named "Mothballs" and notes are always "Deactivated units"

Variable: `unitsInMothballs`

Type: Object

Units Needing Repair: Array of unit identifiers who are crippled.

Variable: `unitsNeedingRepair`

Type: Array

Type	Description
String	{unit designator}+" w/ "+{fleet name}. e.g. "CA w/ Fleet Beta". Units at a colony and not at a fleet substitute {fleet name} for {colony name}.

States of Units: Array of unit identifiers who are have other states.

Variable: `unitStates`

Type: Array

Type	Description
String	{unit designator}+" w/ "+{fleet name}. e.g. "FF w/ Fleet Gunman" Units at a colony and not at a fleet substitute {fleet name} for {colony name}.
String	The state of the unit. Can be "Out of Supply", "Exhausted", "Destroyed", "Gifted". It is never "Crippled" (but see 'unitsNeedingRepair')

9. Other Data

- **Other Empires (`otherEmpires`)**: Array of names of other player empires.
- **Unknown Movement Places (`unknownMovementPlaces`)**: Array of colony names where the statistics are not yet known. These names may not be the actual name of the colony at that location.

Server-Client Considerations

1. **Server-side (PHP):**
    - Use `json_decode()` to read the JS file into PHP associative arrays or objects.
    - Use `json_encode()` to write updates back to this file.
2. **Client-side (JavaScript):**
    - Directly reads the file to display game state and update the UI.
    - All data must be kept in valid JSON-compatible format.

Notes

- All arrays and objects should remain consistent in key naming and type to ensure compatibility between server and client.
- This file serves both as the authoritative game state and configuration reference for gameplay mechanics.
- Colony and unit names must be unique identifiers.
- The key names may be in any order, by it is customarily sorted alphabetically except that `unitList` is placed at the end. This is because `unitList` is generally the largest array and would make the file less readable if other keys are placed after it.
- Standard JSON_PRETTY_PRINT routines are space wasteful. A limited pretty-print algorithm is used: newlines are inserted after closing semicolons and after commas that separate array elements. The exception is that newlines are not inserted into arrays that are themselves nested inside a JSON object.

Specification of the Orders format.

The specific orders that can be found in the orders structure (above) are enumerated as follows. All values are strings, but items noted without quotes represent possible values. For example, currentFlights indicates that the value can be any one of all flight units (units with a design of “LF” or “HF”) owned by this player.

Fleet Deployment orders

Add to Fleet: “type”:”add_fleet”, “receiver”:currentUnits, “target”:"", “note”:“New Fleet Name”
Assign flights: “type”:”flight”, “receiver”:currentFlights, “target”:allBasablePlaces, “note”:””
Rename a fleet: “type”:”name_fleet”, “receiver”:currentFleets, “target”:"", “note”: “New fleet name”

Intelligence Orders

Perform covert mission: “type”:”covert”, “receiver”:colonyNames, “target”:allKnownPlaces, “note”: “Mission Type”
Perform special-forces mission: “type”:”special_force”, “receiver”:colonyNames, “target”:allKnownPlaces, “note”: “Mission Type”

Movement Orders

Convoy Raid: “type”:”convoy_raid”, “receiver”:currentFleets, “target”:allMovablePlaces, “note”: “”
Explore Jump-Lane: “type”:”explore_lane”, “receiver”:currentFleets, “target”:"", “note”: “”
Move fleet: “type”:”move”, “receiver”:currentFleets, “target”:allMovablePlaces, “note”: “”
Load units: “type”:”load”, “receiver”:unitsWithCarry, “target”:allLoadableUnits, “note”: “Amount to Load”
Long-Range Scan: “type”:”long_range”, “receiver”:currentFleets, “target”:allMovablePlaces, “note”: “”
Set a trade route: “type”:”start_trade”, “receiver”:currentFleets, “target”:allKnownPlaces, “note”: “Third system of trade route”
Stop a trade route: “type”:”stop_trade”, “receiver”:currentFleets, “target”:allKnownPlaces, “note”: “”
Unload units: “type”:”unload”, “receiver”:unitsWithCarry, “target”:"", “note”: “Amount to unload”

Diplomatic Orders

Declare War: “type”:”hostile_check”, “receiver”:otherEmpires, “target”:"", “note”: “”
Offer a treaty: “type”:”diplo_check”, “receiver”:otherEmpires, “target”:"", “note”: “”
Sign a treaty: “type”:”sign_treaty”, “receiver”:offeredTreaties, “target”:otherEmpires, “note”: “”
Sneak Attack: “type”;”sneak_attack”, “receiver”:currentFleets, “target”:"", “note”: “”

Construction orders

Build unit at system: “type”:”build_unit”, “receiver”:allBuildableUnits, “target”:colonyNames, “note”: “New fleet name

Convert/Refit Unit: “type”:”convert”, “receiver”:currentUnits, “target”:buildableShips, “note”: “”
Mothball a unit: “type”:”mothball”, “receiver”:currentUnits, “target”:"", “note”: “”
Purchase civilian unit at system: “type”:”purchase_civ”, “receiver”:BuildableCivUnits, “target”:colonyNames, “note”: “New fleet name
Purchase troop at system: “type”:”purchase_troop”, “receiver”:buildableGround, “target”:colonyNames, “note”: “Quantity
Remote build unit: “type”:”remote_build”, “receiver”:buildableBases, “target”:unitsWithCarry, “note”: “”
Repair unit: “type”:”repair”, “receiver”:unitsNeedingRepair, “target”:"", “note”: “”
Scrap a unit: “type”:”scrap”, “receiver”:currentUnits, “target”:"", “note”: “”
Unmothball a unit: “type”:”unmothball”, “receiver”:unitsInMothballs, “target”:"", “note”: “”

Investment Orders

Colonize system: “type”:”colonize”, “receiver”:otherSystems, “target”:"", “note”: “”
Downgrade Lane: “type”:”downgrade_lane”, “receiver”:allKnownPlaces, “target”:allKnownPlaces, “note”: “”
Enact martial law: “type”:”martial_law”, “receiver”:colonyNames, “target”:"", “note”: “”
Improve capacity: “type”:”imp_capacity”, “receiver”:colonyNames, “target”:"", “note”: “”
Improve Population: “type”:”imp_pop”, “receiver”:colonyNames, “target”:"", “note”: “”
Improve Intelligence: “type”:”imp_intel”, “receiver”:colonyNames, “target”:"", “note”: “”
Improve Fortifications: “type”:”imp_fort”, “receiver”:colonyNames, “target”:"", “note”: “”
Invest into research: “type”:”research”, “receiver”:, “target”:"", “note”: “Amount to Invest
(Re) name a colony: “type”:”name_place”, “receiver”:colonyNames, “target”:"", “note”: “Name”
Research Target: “type”:”research_new”, “receiver”:'Research New Unit' || 'Upgrade Unit', “target”:allBuildableUnits, “note”: “”
Upgrade Lane: “type”:”upgrade_lane”, “receiver”:[allKnownPlaces], “target”:[allKnownPlaces], “note”: “”

Combat Orders

Cripple unit: “type”:”cripple”, “receiver”:currentUnits, “target”:"", “note”: “”
Destroy unit: “type”:”destroy”, “receiver”:currentUnits, “target”:"", “note”: “”
Transfer ownership of unit: “type”:”gift”, “receiver”:currentUnits, “target”:otherEmpires, “note”: “”