Data Files - Design

# Overview

This document describes the design of the Stars! Nova Data Files. This design is subordinate to the Stars! Nova Data Files Specification.

The focus of this document is the files which nova creates to communicate turn information and the objects that interact with these. The internal processing of a turn within the server will be documented elsewhere.

# Stars! Files

These are the files the original Stars! uses, and are provided as a comparative reference. Note that Stars! assumed the client would send reliable information (e.g. population) which leads to exploits which would be unacceptable in an open format. Unlike Stars!, nova must be able to verify anything a client tells it. Information in this document regarding these files is based on assumptions about their function (from playing Stars!), not detailed analysis of their contents.

## Game.xy

The initial state of the universe. The server and clients require access to this information.

## Game.hst

The game history file created after the first turn is generated.

## Race.r

The information for each race which is only read when generating the initial game (races can’t change after the beginning of the game.)

## Race.h

History file which records the last known state of each star (owner, pop, minerals, etc.)

## Race.m

A new turn generated by the server, ready for play.

## Race.x

A turn submitted by the client, ready for generation of the next turn.

# Nova Files Overview

Stars! and Stars! Nova use a client/server architecture to support multiplayer LAN and internet (or even play by email) gamming. The Nova GUI (client) provides a means for the player to take turns. The Nova Console (server) combines player orders and information about the game universe to generate subsequent turns. Each application (client & server) has a different internal representation of the game universe, because the server knows everything, whilst the client knows only what the player has explored, and some of that information may be out of date. These representations are built up from game objects (stars, ships, minefields, etc) which are common to both applications. Figure 1 shows an overview of the nova file processing.

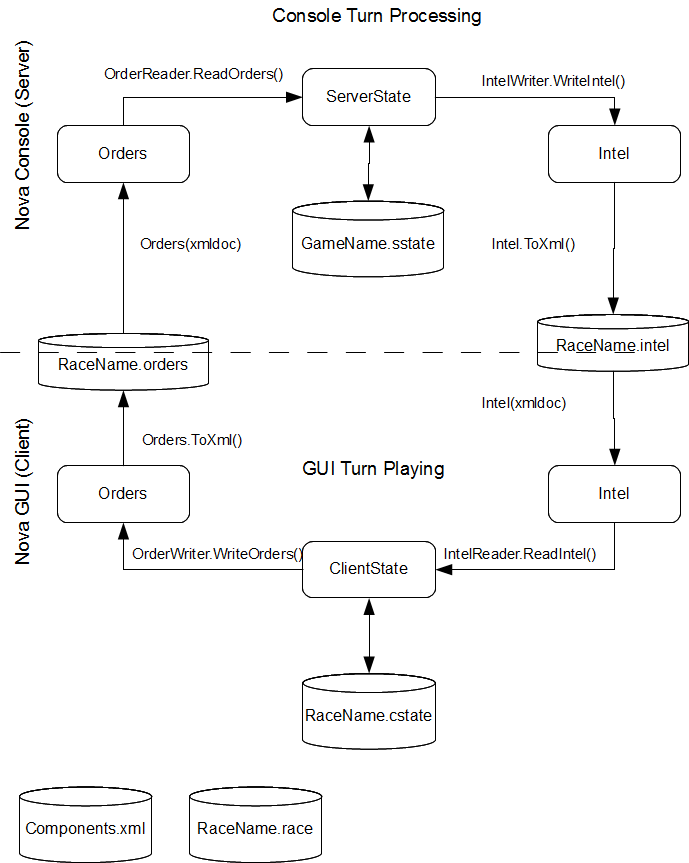


Figure 1—Nova Turn Processing

Nova console (server) uses the ServerState object as its internal representation and can serialise its state into the file <GameName>.sstate which is a persistent representation of the whole game universe, roughly equivalent to the Stars! Game.hst file. The console reads in player orders (if any exist) and updates the ServerState via OrderReader.ReadOrders() in file OrderReader.cs. When the console has processed a turn it generates Intel objects via IntelWriter.WriteIntel() in the file IntelWriter.cs. Serialisation of the Intel objects produces the <RaceName>.intel files. These files contains all the turn data for each player/race. (Note: Currently each of these files contain all the intel for each race. This is bad as it leads to exploits by examining this file, manually or with 3rd party tools, for information not available to the current player’s race. It is planned to reduce the contents of these files to just the information available to the race.) This file can then be read by the nova GUI allowing play of the next turn.

The nova GUI uses the ClientState object defined in ClientState.cs as its internal representation of the game universe. This state is serialised as <RaceName>.cstate (roughly equivalent to the stars! .h file) by ClientState.Save() in ClientState.cs between invocations of the nova GUI. When loading a new turn the ClientState is updated by reading the <RaceName>.intel file via IntelReader.ReadIntel() in IntelReader.cs (initiated by GuiState.Initialize). When a turn has been played, an Orders object is generated by OrderWriter.WriteOrders() in OrderWriter.cs, and serialised by the same function into a <RaceName>.orders file, which can be read by the console.

# Nova Objects

The following objects are involved in the processing of nova turn files and state data.

|  |  |  |  |
| --- | --- | --- | --- |
| **Grouping** | **File (.cs)** | **Object** | **Notes** |
| Common | Race | Race | This module defines all the parameters that define the characteristics of a race. These values are all set in the race designer. This object also manages the loading and saving of race data to a file. |
| Common | Orders | Orders | This module contains the data that is generated by the Nova GUI and passed to the Nova Console so that it can generate the turn for the next year. (sounds like a race.x file) |
| Common | Intel | Intel | This module defines the intel that the nova console (server) sends to the GUI (client), including for the first turn. |
| GUI | ClientState | ClientState | Data used by the GUI that will be persistent across multiple program invocations. |
| GUI | IntelReader | IntelReader | This module is invoked (once) when a new turn has been received from the Nova Console. All the appropriate fields in the GUI state data are updated that are relevant to the player's selected race. |
| GUI | OrderWriter | OrderWriter | This GUI module will generate the player's Orders, which are written to file and sent to the Nova Console so that the turn for the next year can be generated. |
| Console | ServerState | ConsoleState | This file contains data that are persistent across multiple invokations of Nova Console. (It also holds the odd item that doesn't need to be persistent but it's just convenient to keep all "global" data in one place. |
| Console | ProcessTurn | ProcessTurn | Generate a new turn by reading in the player turn files to update the master copy of stars, ships, etc. Then do the processing required to take in the passage of one year of time and, finally, write out the new turn file. |
| Console | GenerateIntel | GenerateIntel | This module converts the console's state into Intel and saves it, thereby generating the next turn to be played. |
| Console | OrderReader | OrderReader | This module processes the reading of race orders. |

# Nova Files

These are notes on the specific files and their contents. File formats are direct object serialisations and as such the format of the files is not defined by any specification or design document at this time. (Note: components.xml and <RaceName>.race have been converted to xml format and their formats require documentation.)

## .sstate

A serialisation of the ServerState object (in file ServerState.cs).

Contains:

public ArrayList AllBattles = new ArrayList();

public Hashtable AllTechLevels = new Hashtable();

public Hashtable AllDesigns = new Hashtable();

public Hashtable AllFleets = new Hashtable();

public Hashtable AllRaceData = new Hashtable(); // Data about the race (traits etc)

public Hashtable AllRaces = new Hashtable(); // Data about the race's relations and battle plans, see RaceData

public Hashtable AllStars = new Hashtable();

public Hashtable AllMinefields = new Hashtable();

public bool GameInProgress = false;

public int FleetID = 1;

public int TurnYear = 2100;

public string GameFolder = null;

public string StatePathName = null;

// Victory conditions (with initial default values)

public EnabledValue PlanetsOwned = new EnabledValue(true, 60);

public EnabledValue TechLevels = new EnabledValue(false, 22);

public EnabledValue NumberOfFields = new EnabledValue(false, 4);

public EnabledValue TotalScore = new EnabledValue(false, 1000);

public EnabledValue SecondPlaceScore = new EnabledValue(false, 0);

public EnabledValue ProductionCapacity = new EnabledValue(false, 1000);

public EnabledValue CapitalShips = new EnabledValue(false, 100);

public EnabledValue HighestScore = new EnabledValue(false, 100);

public int TargetsToMeet = 1;

public int MinimumGameTime = 50;

## .intel

Intel file generated by the Nova Console. A serialisation of Intel (in Intel.cs). This file must be present before the GUI will run (since it contains all sorts of important data such as a list of all stars which is needed to draw the star map).

Note that this file is not read again after the first time a new turn is received until a new turn arrives. (There is no need as once the turn is generated it won't change until the next turn.)

Contains:

public int TurnYear = 2100;

public ArrayList Messages = new ArrayList();

public ArrayList Battles = new ArrayList();

public ArrayList AllRaceNames = new ArrayList();

public ArrayList AllScores = new ArrayList();

public Hashtable RaceIcons = new Hashtable();

public Hashtable AllFleets = new Hashtable();

public Hashtable AllDesigns = new Hashtable();

public Hashtable AllStars = new Hashtable();

public Hashtable AllMinefields = new Hashtable();

## .cstate

The serialised ClientState object (from ClientState.cs), produced by ClientState.Save (same file).

Contains:

public ArrayList DeletedDesigns = new ArrayList();

public ArrayList DeletedFleets = new ArrayList();

public ArrayList Messages = new ArrayList();

public GlobalTurn InputTurn = null;

public Hashtable BattlePlans = new Hashtable();

public Hashtable KnownEnemyDesigns = new Hashtable();

public Hashtable PlayerRelations = new Hashtable();

public Hashtable StarReports = new Hashtable();

public Hashtable AvailableComponents = new Hashtable();

public List<Fleet> PlayerFleets = new List<Fleet>();

public List<Star> PlayerStars = new List<Star>();

public Race RaceData = new Race();

public TechLevel ResearchLevel = new TechLevel();

public TechLevel ResearchResources = new TechLevel();

public bool FirstTurn = true;

public double ResearchAllocation = 0;

public int ResearchBudget = 10;

public int TurnYear = 0;

public string GameFolder = null;

public string RaceName = null;

public string ResearchTopic = "Energy";

## .orders

The data that is created by the Nova GUI and read by the Nova Console. A serialised Orders object (in file Orders.cs) produced by OrderWriter.WriteOrders (in file OrderWriter.cs).

Contains:

public ArrayList RaceFleets = new ArrayList();

public ArrayList RaceDesigns = new ArrayList();

public ArrayList RaceStars = new ArrayList();

public ArrayList DeletedFleets = new ArrayList();

public ArrayList DeletedDesigns = new ArrayList();

public RaceData PlayerData = new RaceData();

public int TechLevel = 0;

## .race

All the information that defines a race before the start of the game. Initially created by the race designer, via the Race object. The Race object also manages reading the race file.

## Components.xml

All the tech components (ship weapons, armour, shields, etc), including ship hulls (but not specific ship designs) and planetary installations (except mines and factories). This data can be visualised and modified using the component editor.