Orb:Connect

Introduction

Orb:Connect is a socket-based communication interface that allows external clients to access Orbiter APIs in a human-readable form. The project is based on the Orbiter OUIPC plugin and the intent of Orb:Connect is to drive things like external "mission control" displays, simpits, and other "read-mostly" types of clients. To that end, read or 'get' methods are assumed in the command structure and only active commands such as "set" and "toggle" are identified specifically. Also, most of the read commands that could have multiple targets (i.e. navmodes, engines, etc.) will return the values of all the appropriate targets with one command in order to try and reduce the number of messages needing to be sent. It was assumed that an engine console client, for example, would more likely be created to display all engine statuses rather than just the left main engine.

Acknowledgements

Thanks to Russ (reverend) Purinton for providing the original OUIPC code from which this work is derived.

Thanks to Brandon (hielor) Bolling for the subscription code and bug fixes.

Thanks to Doug (dbeachy) Beachy for providing the new XRVesselCtrl interface that allows such extensive vessel control and feedback capabilities.

A special thank you to Dr. Martin Schweiger for creating such an extraordinary flight simulator.

Requirements

Orb:Connect 2.0 requires Orbiter 2010-P1. It is *not* backward compatible with Orbiter 2006. To use the XCTL functionality, the vessels must implement the XRCtrlInterface version 2.1

Installation

Simply unzip the package into your Orbiter folder. The software comes with a Java console client that can be used to test connectivity and message formats. The source code is available separately under the GNU Public License (GPL).

Configuration

Initially, Orb:Connect is set up to communicate on port 37777. The installation contains a configuration file in the Config directory (OrbConnect.cfg) in which you can change the service port.

Client configuration

A Java test client comes with the plugin for you to experiment with. If your client is running on the same computer as Orbiter, It is preconfigured to communicate using the default port. Instructions for running the client and reconfiguring it to run on a different computer are in a ClientReadme.txt file in the <orbiter-home>/Doc/OrbConnect directory.

Message Structure

This section describes the format of all messages supported by OrbConnect. In support of my goal of maintaining a human-readable and maintainable message structure, commands are broken up into sections, with each section separated by a colon:

Type. What type of message is it? Types are:

- ORB – Messages to Orbiter, or the simulation as a whole.
- CAMERA – Camera controls.
- SHIP – Messages to vessels.
- FOCUS – Convenience messages to operate on the focus vessel.
- XRCTL – Messages to vessels that implement the XRVesselCtrl interface.
- NAV – Messages relating to position and navigation information.
- BODY – Messages to planets, moons, and other 'bodies'.
- OBJ – Messages for handling generic objects.

Target. For commands that target a specific item in Orbiter. Targets may be identified by name or index. A target reference may also have more than one part (specifically bases and base attributes.) If it does, each part of the target will also be separated by colons.

Command.The identifier of the message to send.

Arguments. Any arguments the message needs to fulfill its task. Mainly used in 'SET' messages.

Again, the sections are each separated by a colon (:) and a specific command *may not use all sections*. Within the arguments section, for commands that take multiple arguments, each argument is separated by a comma (,). Note that although I tried to keep the messages human-readable, some of the commands and even more of the responses can be very complex. Take special note of the delimiters if you have trouble – that's the most often source of errors.

Commands are *not* case sensitive. SHIP, Ship, and ship will all work. It is recommended, however, that you use a form of camel case (http://en.wikipedia.org/wiki/CamelCase) for readability and to prevent misspellings. Note, however, that names *are* case sensitive. While "Earth" will be found, "earth" will not. This applies to bodies as well as vessel names. If unsure of a name, consult the scenario configuration file.

Response Structure

The return message or response from Orb:Connect is prefaced by the command that was sent to it, followed by an equal siqn (=) and the actual response. This is to help you ensure multiple responses don't get mixed up.

Return values can be string, integer, decimal, boolean, or lists of any of those. If an API method does not return a value (has a void return type), "OK"; will be returned if successful. Multiple return values are separated by commas (,).

Boolean values are returned as ";1"; for true and ";0"; for false.

Vectors (VECTOR3) are represented as a comma-separated list of x,y,z doubles.

Matrices (MATRIX3) values are returned as a comma separated list in the row/column order 11,12,13,21,22,23,31,32,33.

Structured data returned from an API method will contain the values in the order they appear in the struct. If an array of structs or values is embedded, they will be returned inside brackets ([ ]) and semicolon separated. The use of SHIP:Status2 message where the engine, propellant and docking port properties may be specified for return is a prime example.

In some cases, a list of multiple values may be returned in order to consolidate the same information about multiple entities (i.e. engines) and reduce the amount of communication. In these cases, the list entries are separated by semicolon (;).

Example:

XCTL:Focus:DoorsPos may return 3,0;2,0.5; . . . ;3,1;5,0
where each pair of numbers stands for a separate door instance
state and position: ";3,0"; for door 0, ";2, 0.5"; for door 1, etc.

Another style of a return value containing multiple instances of the same information uses positional notation. This format is used when the information consists of a single character value (i.e. Boolean or states). Using this style, the positions are numbered starting with 0 and each position in the return string stands for the object with that (zero-based) index or enum value.

Example:

SHIP:1:NavModeStates may return 01000000000
This indicates only navmode 1 (KILLROT) of the seven navmodes defined by the OrbiterAPI and the three autopilots defined by the XRVesselCtrl interface is engaged

For additional information, look at the indicated reference API method arguments and return values. Pay close attention to the static values and struct definitions provided by the OrbiterAPI and XRVesselCtrl as these not only provide the *type* of a return value, they also indicate the *order* of return values when the messages return multiple values. You are strongly encouraged to use the test client to view complex responses.

Errors

Serious errors that may preclude the plugin from working are logged in the orbiter.log file. Command errors are identified by a return value beginning with "ERR" (e.g. ERR01). The descriptions of the error codes are contained in a later section.

Message Index

Below is a list of OrbConnect messages. The section is divided into subsections that contain messages for specific 'areas' of Orbiter control and information. Commands are ordered by area, and are generally listed in the order found in the Orbiter SDK Reference Manual. The listing uses angle brackets (< .. >) to delineate a variable argument (something you must provide) that is required prior to the command, with a descriptive phrase inside to indicate its content. That content is entered into the command without the brackets. Argument values are separated by commas.

Example:

ORB:Name:<object index> would actually be created as **ORB:Name:2** to retrieve the name of the object with the index of 2.

Subscription Messages

These messages provide the ability to set up repetitive data transmissions without having to continually send commands. Subscribe messages consist of three parts: the command, an update rate/frequency, and the target orbiter command. The subscription command returns a unique number that allows the client to identify the incoming message and associate arriving data back to the subscription command.

Unsubscribe messages consist of two parts: the command and the subscription id returned by the subscribe command. It is always good practice for clients to unsubscribe from each subscription prior to exiting or closing connections.

Update rates are set in number of updates per second, the term Hertz (Hz) is commonly used for this rate. The service divides a second into 20 timeslots, which allow for rates of 1 to 20Hz. While you can use an arbitrary value as the frequency, the problem is data can only be sent out during a timeslot --so the data will "stutter" because it will be alternating between the expected and one additional timeslot. This problem is avoided when you use a "native" frequency, which in the case of 20 buckets will be 20, 10, 6.6667, 5, 4, 3.333, 2.85...etc Hz. All the native frequencies can be calculated as 20/timeslot where timeslot is an integer between 1 and 20.

Clients may subscribe to any number of commands. Note, however, that large numbers of subscriptions at high refresh rates may negatively impact frame rates. Here’s an example to return the focus vessels altitude four times per second:

```
Client:  SUBSCRIBE:4:SHIP:FOCUS:Alt
Orbiter: SUBSCRIBE:4:SHIP:FOCUS:Alt=1000
         1000=1200
         1000=1201
         1000=1202
         . . .
Client:  UNSUBSCRIBE:1000
Orbiter: UNSUBSCRIBE:1000=OK
```

SUBSCRIBE:<frequency>:<command>

Reference API method	None
Additional Arguments	None
Return Type	A unique subscription id (int)

UNSUBSCRIBE<subscriptionId>

Reference API method	None
Additional Arguments	None
Return Type	"OK"

General Orbiter Messages

These messages provide information about the active simulation, as well as control over some of its properties. Reference API is OrbiterAPI.

ORB:GBodyCount

Reference API method	oapiGetGBodyCount
Additional Arguments	None
Return Type	int

ORB:GBodies

Reference API method	none
Additional Arguments	None
Return Type	csv list of GBodies defined in the current scenario Moons of a planet are prefixed by the primaries name: Earth.Moon, Saturn.Titan, etc.

ORB:SimTime

Reference API method	oapiGetSimTime
Additional Arguments	None
Return Type	double

ORB:SimStep

Reference API method	oapiGetSimStep
Additional Arguments	None
Return Type	double

ORB:SysTime

Reference API method	oapiGetSysTime
Additional Arguments	None
Return Type	double

ORB:SysStep

Reference API method	oapiGetSysStep
Additional Arguments	None
Return Type	double

ORB:SimMJD

Reference API method	oapiGetSimMJD
Additional Arguments	None
Return Type	double

ORB:SysMJD

Reference API method	oapiGetSysMJD
Additional Arguments	None
Return Type	double

ORB:SetSimMJD

Reference API method	oapiSetSimMJD
Additional Arguments	MJD (double)
Return Type	double

ORB:Time2MJD

Reference API method	oapiTime2MJD
Additional Arguments	Time (double)
Return Type	double

ORB:TimeAccel

Reference API method	oapiGetTimeAcceleration
Additional Arguments	None
Return Type	double

ORB:SetTimeAccel

Reference API method	oapiSetTimeAcceleration
Additional Arguments	Warp (double)
Return Type	"OK"

ORB:Pause

Reference API method	oapiGetPause
Additional Arguments	None
Return Type	bool

ORB:SetPause

Reference API method	oapiSetPause
Additional Arguments	Paused (bool)
Return Type	"OK"

ORB:FrameRate

Reference API method	oapiGeFrameRate
Additional Arguments	None
Return Type	double

ORB:HUDMode

Reference API method	oapiGetVesselCount
Additional Arguments	None
Return Type	int

ORB:SetHUDMode

Reference API method	oapiSetHUDMode
Additional Arguments	mode (int)
Return Type	bool

ORB:ToggleHUDColor

Reference API method	oapiToggleHUDColour
Additional Arguments	None
Return Type	"OK"

ORB:IncHUDIntensity

Reference API method	oapiIncHUDIntensity
Additional Arguments	None
Return Type	"OK"

ORB:DecHUDIntensity

Reference API method	oapiDecHUDIntensity
Additional Arguments	None
Return Type	"OK"

ORB:SetHUDBrightness

Reference API method	None. Sets HUD brightness directly.
Additional Arguments	a brightness level between 0 and 100 (int)
Return Type	"OK"

ORB:MouseClick

Reference API method	N/A - performs mouse click at specified coords
Additional Arguments	screen x coord (int) screen y coord (int)
Return Type	"OK"

ORB:OpenMFD

Reference API method	oapiOpenMFD
Additional Arguments	id (int) mode (int)
Return Type	int

ORB:MFDMode

Reference API method	oapiGetMFDMode
Additional Arguments	id (int)
Return Type	int

ORB:SendMFDKey

Reference API method	oapiSendMFDKey
Additional Arguments	id (int) key code (int)
Return Type	int
Notes	MFD must be in SEL or MNU mode or keys are ignored SEL btn = OAPI_KEY_F1 MNU btn = OAPI_KEY_GRAVE

ORB:ProcessMFDButton

Reference API method	oapiProcessMFDButton
Additional Arguments	id (int) button (int) mouse event (int)
Return Type	bool

ORB:MFDButtonLabel

Reference API method	oapiMFDButtonLabel
Additional Arguments	id (int) button (int)
Return Type	string

ORB:MFDButtonLabels

Reference API method	oapiMFDButtonLabel
Additional Arguments	id (int)
Return Type	Comma-separated string of button labels

ORB:VirtualMFDs

Reference API method	VNCMFD:GetMfdIds()
Additional Arguments	none
Return Type	Comma-separated string of VNCMFD ids

ORB:SwitchPanel

Reference API method	oapiSwitchPanel
Additional Arguments	direction (int)
Return Type	int

ORB:SetPanel

Reference API method	oapiSetPanel
Additional Arguments	panelId (int)
Return Type	int

ORB:DebugString

Reference API method	oapiDebugString
Additional Arguments	"CLEAR" or a message
Return Type	"OK"

Camera Messages

These messages allow for positioning and control over the Orbiter camera view. Reference API is OrbiterAPI.

CAMERA:IsInternal

Reference API method	oapiCameraInternal
Additional Arguments	None
Return Type	bool

CAMERA:Mode

Reference API method	oapiCameraMode
Additional Arguments	None
Return Type	int

CAMERA:CockpitMode

Reference API method	oapiCockpitMode
Additional Arguments	None
Return Type	int

CAMERA:Target

Reference API method	oapiCameraTarget
Additional Arguments	None
Return Type	string (name of target object)

CAMERA:GlobalPos

Reference API method	oapiCameraGlobalPos
Additional Arguments	None
Return Type	vector

CAMERA:GlobalDir

Reference API method	oapiCameraGlobalDir
Additional Arguments	None
Return Type	vector

CAMERA:TargetDist

Reference API method	oapiCameraTargetDist
Additional Arguments	None
Return Type	double

CAMERA:Azimuth

Reference API method	oapiCameraAzimuth
Additional Arguments	None
Return Type	double

CAMERA:Polar

Reference API method	oapiCameraPolar
Additional Arguments	None
Return Type	double

CAMERA:Aperture

Reference API method	oapiCameraAperture
Additional Arguments	None
Return Type	double

CAMERA:SetAperture

Reference API method	oapiCameraSetAperture
Additional Arguments	aperture (double)
Return Type	"OK"

CAMERA:SetScaleDist

Reference API method	oapiCameraScaleDist
Additional Arguments	scale factor (double)
Return Type	"OK"

CAMERA:RotAzimuth

Reference API method	oapiCameraRotAzimuth
Additional Arguments	azimuth change (double)
Return Type	"OK"

CAMERA:RotPolar

Reference API method	oapiCameraRotPolar
Additional Arguments	polar change (double)
Return Type	"OK"

CAMERA:SetCockpitDir

Reference API method	oapiCameraSetCockpitDir
Additional Arguments	polar dir (double)
	azimuth dir (double)
Return Type	"OK"

Vessel Related Messages

The following messages are for vessel information/handling. These commands normally take an object identifier (name or index) before the command name. The special identifier "FOCUS" may be used to reference the vessel that has focus. Reference API is OrbiterAPI if the reference method begins with "oapi", otherwise the reference is to the VESSEL/VESSEL2 interface.

SHIP:Count

Reference API method	oapiGetVesselCount
Additional Arguments	None
Return Type	int

SHIP:CockpitMode

Reference API method	oapiCockpitMode
Additional Arguments	None
Return Type	bool

SHIP:<"FOCUS" or vessel index>:Name

Reference API method	GetName
Additional Arguments	None
Return Type	string

SHIP:<"FOCUS", vessel name or index>:ClassName

Reference API method	GetClassName
Additional Arguments	None
Return Type	string

SHIP:<"FOCUS", vessel name or index>:Version

Reference API method	Version
Additional Arguments	None
Return Type	int

SHIP:<"FOCUS", vessel name or index>:Status

Reference API method	GetStatus
Additional Arguments	None
Return Type	VESSELSTATUS as csv

SHIP:<"FOCUS", vessel name or index>:Status2

Reference API method	GetStatusEx
Additional Arguments	get fuel levels (bool) OPTIONAL get thruster levels (bool) OPTIONAL get docking info (bool) OPTIONAL
Return Type	VESSELSTATUS2 as csv semicolon fuel levels for 0 <= # fuelspecs as csv (if requested) semicolon thruster levels for 0 <= # thrusters as csv (if requested) semicolon ref vessel docking port, ref vessel name for 0 <= # dockinfos as csv (if requested)

SHIP:<"FOCUS", vessel name or index>:Mass

Reference API method	GetMass
Additional Arguments	None
Return Type	double

SHIP:<"FOCUS", vessel name or index>:EmptyMass

Reference API method	GetEmptyMass
Additional Arguments	None
Return Type	double

SHIP:<"FOCUS", vessel name or index>:PropMass

Reference API method	GetPropellantMass
Additional Arguments	None
Return Type	double

SHIP:<"FOCUS", vessel name or index>:PropFlowRate

Reference API method	GetPropellantFlowrate
Additional Arguments	None
Return Type	double

SHIP:<"FOCUS", vessel name or index>:DfltFuelMass

Reference API method	GetFuelMass
Additional Arguments	None
Return Type	double

SHIP:<"FOCUS", vessel name or index>:DfltMaxFuelMass

Reference API method	GetMaxFuelMass
Additional Arguments	None
Return Type	double

SHIP:<"FOCUS", vessel name or index>:DfltFuelFlowRate

Reference API method	GetFuelRate
Additional Arguments	None
Return Type	double

SHIP:<"FOCUS", vessel name or index>:Alt

Reference API method	GetAltitude
Additional Arguments	None
Return Type	double

SHIP:<"FOCUS", vessel name or index>:NavModeStates

Reference API method	GetNavmodeState
Additional Arguments	None
Return Type	positional string of NavModes (AutoPilots) by id (1-based) The three XR autopilots are included in the last three positions: AttitudeHold, DescentHold and AirspeedHold.

SHIP:<"FOCUS", vessel name or index>:SetNavMode

Reference API method	ActivateNavmode, DeactivateNavmode
Additional Arguments	id (int) activate (bool)
Return Type	bool

SHIP:<"FOCUS", vessel name or index>:ToggleNavmode

Reference API method	ToggleNavmode
Additional Arguments	None
Return Type	int

SHIP:<"FOCUS", vessel name or index>:APSettings

Reference API method	None
Additional Arguments	None
Return Type	Comma separated values of the flight variables set (held) by the NavModes/Autopilots. They are in pairs of active (bool) and value (double) and appear in the order: Bank, Pitch, AoA, Altitude, Airspeed, Vertical Speed (bool,double,bool,double, bool,double,bool,double, bool,double,bool,double)

SHIP:<"FOCUS", vessel name or index>:AttitudeMode

Reference API method	oapiGetAttitudeMode
Additional Arguments	None
Return Type	int

SHIP:<"FOCUS", vessel name or index>:SetAttitudeMode

Reference API method	oapiSetAttitudeMode
Additional Arguments	mode (int)
Return Type	bool

SHIP:<"FOCUS", vessel name or index>:ToggleAttitudeMode

Reference API method	oapiToggleAttitudeMode
Additional Arguments	None
Return Type	int

SHIP:<"FOCUS", vessel name or index>:ADCtrlMode

Reference API method	GetADCtrlMode
Additional Arguments	None
Return Type	int

SHIP:<"FOCUS", vessel name or index>:SetADCtrlMode

Reference API method	SetADCtrlMode
Additional Arguments	mode (int)
Return Type	"OK"

SHIP:<"FOCUS", vessel name or index>:EngineGrpLevels

Reference API method	GetThrusterGroupLevel(2)
Additional Arguments	None
Return Type	csv of levels (double) for each thruster group

SHIP:<"FOCUS", vessel name or index>:SetEngineGrpLevel

Reference API method	SetThrusterGroupLevel(2)
Additional Arguments	groupId (int) level (double)
Return Type	"OK"

SHIP:<"FOCUS", vessel name or index>:ChgEngineGrpLevel

Reference API method	IncThrusterGroupLevel(2)
Additional Arguments	groupId (int) level change (double)
Return Type	"OK"

SHIP:<"FOCUS", vessel name or index>:FltStatus

Reference API method	GetFlightStatus
Additional Arguments	None
Return Type	int

SHIP:<"FOCUS", vessel name or index>:Airspd

Reference API method	GetAirspeed
Additional Arguments	None
Return Type	double

SHIP:<"FOCUS", vessel name or index>:TruSpd (Alias for Airspd message)

Reference API method	GetAirspeed (See SurfaceMFD in Orbiter manual)
Additional Arguments	None
Return Type	double

SHIP:<"FOCUS", vessel name or index>:ShipAirspdVector

Reference API method	GetShipAirspeedVector
Additional Arguments	None
Return Type	vector

SHIP:<"FOCUS", vessel name or index>:Accel

Reference API method	None
Additional Arguments	None
Return Type	The ship acceleration in m/s^2 along the airspeed vector (double)

SHIP:<"FOCUS", vessel name or index>:VAccel

Reference API method	None
Additional Arguments	None
Return Type	The ship vertical acceleration in m/s^2 (double)

SHIP:<"FOCUS", vessel name or index>:IndSpd

Reference API method	None Returns the ‘indicated’ airspeed based on atmospheric conditions and flight regime. (See SurfaceMFD in Orbiter manual)
Additional Arguments	None
Return Type	double

SHIP:<"FOCUS", vessel name or index>:OrbSpd

Reference API method	None Returns the orbital speed of the vessel (See SurfaceMFD in Orbiter manual)
Additional Arguments	None
Return Type	double

SHIP:<"FOCUS", vessel name or index>:GndSpd

Reference API method	None Returns the ground speed of the vessel (See SurfaceMFD in Orbiter manual)
Additional Arguments	None
Return Type	double

SHIP:<"FOCUS", vessel name or index>:EquSpd

Reference API method	None Returns the ‘equivalent’ airspeed based on atmospheric conditions and flight regime. (See SurfaceMFD in Orbiter manual)
Additional Arguments	None
Return Type	double

SHIP:<"FOCUS", vessel name or index>:HorizonAirspdVector

Reference API method	GetHorizonAirspeedVector
Additional Arguments	None
Return Type	vector

SHIP:<"FOCUS", vessel name or index>:Attitude

Reference API method	GetAoA, GetSlipAngle, GetPitch, GetBank
Additional Arguments	None
Return Type	AoA, slip, pitch, bank(double,double,double,double)

SHIP:<"FOCUS", vessel name or index>:AtmConditions

Reference API method	GetAtmTemperature, GetAtmDensity, GetAtmPressure, GetDynPressure, GetMachNumber
Additional Arguments	None
Return Type	temp,density,pressure,dynamic pressure, mach nbr (double,double,double,double,double)

Focus Object Messages

The below messages allow operations on the focus vessel via the specific Orbiter methods available for that purpose. They are a slightly shorter version of the SHIP: messages, but of course will not track a specific vessel if the focus changes.

FOCUS:Name

Reference API method	oapiGetFocusObjectName
Additional Arguments	None
Return Type	string

FOCUS:SetbyIndex

Reference API method	oapiSetFocusObject, oapiGetVesselByIndex
Additional Arguments	index (int)
Return Type	"OK"

FOCUS:SetByName

Reference API method	oapiSetFocusObject, oapiGetVesselByName
Additional Arguments	name (string)
Return Type	"OK"

FOCUS:GlobalPos

Reference API method	oapiGetFocusGlobalPos
Additional Arguments	None
Return Type	vector

FOCUS:GlobalVel

Reference API method	oapiGetFocusVesselVel
Additional Arguments	None
Return Type	vector

FOCUS:RelPos

Reference API method	oapiGetFocusRelativePos
Additional Arguments	reference object id (name or object index)
Return Type	vector
	Reference Frame (int) Optional
	0=Global (default)
	1=Vessel Local (see Global2Local)
	2=Vessel Horizon (see HorizonRot)

FOCUS:RelVel

Reference API method	oapiGetFocusRelativeVel
Additional Arguments	reference object id (name or object index)
Return Type	vector
	Reference Frame (int) Optional
	0=Global (default)
	1=Vessel Local (see Global2Local)
	2=Vessel Horizon (see HorizonRot)

FOCUS:RelPosVel

Reference API method	oapiGetFocusRelativePos oapiGetFocusRelativeVel
Additional Arguments	reference object id (name or object index) Reference Frame (int) Optional 0=Global (default) 1=Vessel Local (see Global2Local) 2=Vessel Horizon (see HorizonRot)
Return Type	vector,vector

FOCUS:Alt

Reference API method	oapiGetFocusAltitude
Additional Arguments	None
Return Type	double

FOCUS:Pitch

Reference API method	oapiGetFocusPitch
Additional Arguments	None
Return Type	double

FOCUS:Bank

Reference API method	oapiGetFocusBank
Additional Arguments	None
Return Type	double

FOCUS:Heading

Reference API method	oapiGetFocusHeading
Additional Arguments	None
Return Type	double

FOCUS:EquPos

Reference API method	oapiGetFocusEquPos
Additional Arguments	None
Return Type	vector

FOCUS:Airspd

Reference API method	oapiGetFocusAirspeed
Additional Arguments	None
Return Type	double

FOCUS:AirspdVector

Reference API method	oapiGetFocusAirspeedVector
Additional Arguments	None
Return Type	double

FOCUS:ShipAirspdVector

Reference API method	oapiGetFocusShipAirspeedVector
Additional Arguments	None
Return Type	double

FOCUS:AtmDensity

Reference API method	oapiGetFocusAtmPressureDensity
Additional Arguments	None
Return Type	pressure, density (double,double)

FOCUS:EngineStatus

Reference API method	oapiGetFocusEngineStatus
Additional Arguments	None
Return Type	mainLevel, hoverLevel, attMode (double,double,int)

FOCUS:AttitudeMode

Reference API method	oapiGetFocusAttitudeMode
Additional Arguments	None
Return Type	int

FOCUS:ToggleAttitudeMode

Reference API method	oapiToggleFocusAttitudeMode
Additional Arguments	None
Return Type	int

FOCUS:SetAttitudeMode

Reference API method	oapiSetFocusAttitudeMode
Additional Arguments	None
Return Type	"OK"

XRVesselCtrl Interface Specific Messages

These commands are for accessing extended commands available through the XRVesselCtrl interface created by Doug Beachy. This is a public interface that was created initially for his XR- series vessels, but it may be implemented by any vessel addon developer. These commands take a *vessel* identifier (name or index) before the command name in the same manner as SHIP messages.

XCTL:<"FOCUS", vessel name or index>:XRVersion

Reference API method	GetCtrlApiVersion
Additional Arguments	None
Return Type	double

XCTL:<"FOCUS", vessel name or index>:SetEngine

Reference API method	SetEngineState
Additional Arguments	engineId (int)
	throttle level (double)
	pitch gimbal position (double)
	yaw gimbal position (double)
	balance position (double)
	pitch centering mode engaged (bool)
	yaw centering mode engaged (bool)
	balance centering mode engaged (bool)
	auto mode engaged (bool)
	divergent mode engaged (bool)
Return Type	"OK"

XCTL:<"FOCUS", vessel name or index>:Engine

Reference API method	GetEngineState
Additional Arguments	engineId (int)
Return Type	engineId (int),
	throttle level (double),
	pitch gimbal position (double),
	yaw gimbal position (double),
	balance position (double),
	pitch centering mode engaged (bool),
	yaw centering mode engaged (bool),
	balance centering mode engaged (bool),
	auto mode engaged (bool),
	divergent mode engaged (bool),
	tsfc (double),
	flow rate (double),
	thrust (double),
	fuel level (double),
	max fuel level (double),
	diffuser temp (double),
	burner temp (double),
	exhaust temp (double)

XCTL:<"FOCUS", vessel name or index>:Engines

Reference API method	GetEngineState
Additional Arguments	None
Return Type	semicolon separated string of engines states by engineId (see Engine above)

XCTL:<"FOCUS", vessel name or index>:Doors

Reference API method	GetDoorState
Additional Arguments	None
Return Type	semicolon separated string of (int) door states by doorId, plus the IsElevatorActiveEVA state (0=false, 1=true).

XCTL:<"FOCUS", vessel name or index>:DoorsPos

Reference API method	GetDoorState
Additional Arguments	None
Return Type	semicolon separated string of (comma separated door State (int) and Position (double)) by doorId

XCTL:<"FOCUS", vessel name or index>:SetDoor

Reference API method	SetDoorState
Additional Arguments	doorId (int)
	doorState (int)
Return Type	"OK"

XCTL:<"FOCUS", vessel name or index>:TglDoor

Reference API method	None – toggles the door state between open[ing] and clos[ing]
Additional Arguments	doorId (int)
Return Type	"OK"

XCTL:<"FOCUS", vessel name or index>:KillAPilots

Reference API method	KillAutopilots
Additional Arguments	None
Return Type	"OK"

XCTL:<"FOCUS", vessel name or index>:SetFreeDrift

Reference API method	KillAutopilots, SetADCtrlMode, SetAttitudeMode (turns all off)
Additional Arguments	None
Return Type	"OK"

XCTL:<"FOCUS", vessel name or index>:StdAPs

Reference API method	GetStandardAP
Additional Arguments	None
Return Type	positionalString of standard autopilots (Navmodes) by id

XCTL:<"FOCUS", vessel name or index>:SetStdAP

Reference API method	SetStandardAP
Additional Arguments	autopilotId/navmode (int)
	engaged (bool)
Return Type	bool

XCTL:<"FOCUS", vessel name or index>:AttHldAP

Reference API method	GetAttitudeHoldAP
Additional Arguments	None
Return Type	engaged (bool),
	mode (int),
	targetPitch (double),
	targetBank (double)

XCTL:<"FOCUS", vessel name or index>:SetAttHldAP

Reference API method	SetAttitudeHoldAP
Additional Arguments	engaged (bool), mode (int), targetPitch (double), targetBank (double)
Return Type	int

XCTL:<"FOCUS", vessel name or index>:ExpAPs (Expanded Autopilots)

Reference API method	GetStandardAP
Additional Arguments	None
Return Type	positionalString of autopilot configurations (Navmodes) (engaged = 0, disengaged = 1) Positions: - KillRot - Attitude hold on with pitch and bank = 0 (mimics standard LVLH) - Prograde - Retrograde - Normal - AntiNormal - Descent Hold - Hover (Descent Hold on, target rate of 0.0, not in AutoLand mode) - Autoland (Descent Hold on, in AutoLand mode) - Inertial Attitude (Attitude Hold on, with non-level attitude. i.e. pitch and/or bank not = 0. - Airspeed Hold - Free Drift (No autopilots engaged, Aero surfaces and RCS both OFF). Does not take engines into account, so you may be pushing something with it in control of the attitude of the stack.

XCTL:<"FOCUS", vessel name or index>:ToggleAttHldAP (toggle on/off state of Attitude Hold AP, w/o changing targets)

Reference API method	SetAttitudeHoldAP
Additional Arguments	none
Return Type	int

XCTL:<"FOCUS", vessel name or index>:ToggleCurrAttHldAP (Toggles on off state of Attitude Hold A/P. When A/P is disengaged, targets are left as-is. When A/P is engaged, it syncs the targets to the current vessel attitude.)

Reference API method	None
Additional Arguments	None
Return Type	int

XCTL:<"FOCUS", vessel name or index>:ToggleLVLHAP (Toggles on off state of Attitude Hold A/P. When A/P is disengaged, targets are left as-is. When A/P is engaged, it zeroes the target pitch and bank)

Reference API method	None
Additional Arguments	engaged (bool), mode (int), targetPitch (double), targetBank (double)
Return Type	int

XCTL:<"FOCUS", vessel name or index>:DscntHldAP

Reference API method	GetDescentHoldAP
Additional Arguments	None
Return Type	engaged (bool),
	targetVerticalSpeed (double),
	autoland (bool)

XCTL:<"FOCUS", vessel name or index>:SetDscntHldAP

Reference API method	oapiGetVesselCount
Additional Arguments	engaged (bool),
	targetVerticalSpeed (double),
	autoland (bool)
Return Type	int

XCTL:<"FOCUS", vessel name or index>:AirspdAP

Reference API method	GetAirspeedHoldAP
Additional Arguments	None
Return Type	engaged (bool),
	targetAirspeed (double),

XCTL:<"FOCUS", vessel name or index>:SetAirspdAP

Reference API method	SetAirspeedHoldAP
Additional Arguments	engaged (bool),
	targetAirspeed (double),
Return Type	int

XCTL:<"FOCUS", vessel name or index>:SysStatus

Reference API method	GetXRSystemStatus
Additional Arguments	None
Return Type	csv string of XRSystemStatus values

XCTL:<"FOCUS", vessel name or index>:MWS

Reference API method	GetXRSystemStatus
Additional Arguments	None
Return Type	positional string of bools by XRSystemStatus value.
	"1" (true) indicates a status value < 1.0 or not "offline".

XCTL:<"FOCUS", vessel name or index>:ResetMWS

Reference API method	ResetMWS
Additional Arguments	None
Return Type	int

XCTL:<"FOCUS", vessel name or index>:ExtLights

Reference API method	GetExteriorLight
Additional Arguments	None
Return Type	comma separated bools by lightId

XCTL:<"FOCUS", vessel name or index>:SetExtLight

Reference API method	SetExteriorLight
Additional Arguments	lightId (int)
	on (bool)
Return Type	bool

XCTL:<"FOCUS", vessel name or index>:HUDModes

Reference API method	oapiGetHUDMode, GetSecondaryHUDMode, GetTertiaryHUDMode
Additional Arguments	None
Return Type	priMode, secMode, terMode (int,int,int) Note: priMode will be empty if vessel does not have focus.

XCTL:<"FOCUS", vessel name or index>:SetHUDMode

Reference API method	oapiSetHUDMode, SetSecondaryHUDMode, SetTertiaryHUDMode
Additional Arguments	HUDId (int) pri=1, sec=2, tert=3 mode (int)
Return Type	bool

XCTL:<"FOCUS", vessel name or index>:CoG

Reference API method	GetCenterOfGravity
Additional Arguments	None
Return Type	double

XCTL:<"FOCUS", vessel name or index>:ChgCoG

Reference API method	ShiftCenterOfGravity
Additional Arguments	meters to shift (double)
Return Type	bool

XCTL:<"FOCUS", vessel name or index>:RCSDockingMode

Reference API method	IsRCSDockingMode
Additional Arguments	None
Return Type	bool

XCTL:<"FOCUS", vessel name or index>:SetRCSDockingMode

Reference API method	SetRCSDockingMode
Additional Arguments	bool
Return Type	bool

XCTL:<"FOCUS", vessel name or index>:ElevEVA

Reference API method	IsElevatorEVAPortActive
Additional Arguments	None
Return Type	bool

XCTL:<"FOCUS", vessel name or index>:SetElevEVA

Reference API method	SetElevatorEVAPortActive
Additional Arguments	bool
Return Type	bool

XCTL:<"FOCUS", vessel name or index>:StatusMsgs

Reference API method	GetStatusScreenText
Additional Arguments	nbr of lines requested (int) OPTIONAL if not specified, all available messages (up to 64) are returned.
Return Type	Semicolon separated list of status messages from Tertiary HUD

Position and Navigation Messages

These messages return information about a vessels position and velocity. This information can be absolute or relative to other vessels, planetary bodies, or even navaids, Different reference frames may also be requested. Reference API is OrbiterAPI.

NAV:<"FOCUS", vessel name or index>:Elements1

Reference API method	GetElements(1)
Additional Arguments	None
Return Type	ELEMENTS struct as csv

NAV:<"FOCUS", vessel name or index>:Elements2

Reference API method	GetElements(2)
Additional Arguments	reference body name (optional)
	mjd (double) (optional)
	Note: Accepts “now” to use current simMJD
	frame (int) (optional)
Return Type	ELEMENTS then ORBITPARAM structs as csv,

NAV:<"FOCUS", vessel name or index>:EquPos

Reference API method	GetEquPos
Additional Arguments	None
Return Type	long (double), lat (double), rad (double)

NAV:<"FOCUS", vessel name or index>:GravRef

Reference API method	GetGravityRef
Additional Arguments	None
Return Type	String

NAV:<"FOCUS", vessel name or index>:GlobalPos

Reference API method	GetGlobalPos
Additional Arguments	None
Return Type	vector

NAV:<"FOCUS", vessel name or index>:GlobalVel

Reference API method	GetGlobalVel
Additional Arguments	None
Return Type	vector

NAV:<"FOCUS", vessel name or index>:RelPos

Reference API method	oapiGetFocusRelativePos
Additional Arguments	reference object id (name or object index)
Return Type	vector
	Reference Frame (int) Optional
	0=Global (default)
	1=Vessel Local (see Global2Local)
	2=Vessel Horizon (see HorizonRot)

NAV:<"FOCUS", vessel name or index>:RelVel

Reference API method	oapiGetFocusRelativeVel
Additional Arguments	reference object id (name or object index)
Return Type	vector
	Reference Frame (int) Optional 0=Global (default) 1=Vessel Local (see Global2Local) 2=Vessel Horizon (see HorizonRot)

NAV:<"FOCUS", vessel name or index>:RelPosVel

Reference API method	oapiGetFocusRelativePos oapiGetFocusRelativeVel
Additional Arguments	reference object id (name or object index) Reference Frame (int) Optional 0=Global (default) 1=Vessel Local (see Global2Local) 2=Vessel Horizon (see HorizonRot)
Return Type	vector,vector

NAV:<"FOCUS", vessel name or index>:Navaids

Reference API method	None. Returns a list of nav transmitters
Additional Arguments	Type of transmitter (int). Optional. Returns all types by default. Limit (int). Optional. Number of navaids to return. Default is 100. Sort by name (bool). Optional. Default is sort by distance.
Return Type	Semicolon delimited list of csv navaid info: Type (int), Description (string), Channel (int), InRange (bool)

NAV:<"FOCUS", vessel name or index>:NavaidRelPos

Reference API method	None. Returns information about navaid position, distance and bearing if in range. Options include different reference frames including vessel, horizon relative
Additional Arguments	Receiver number Reference Frame (int) Optional 0=Global (default) 1=Vessel Local (see Global2Local) 2=Vessel Horizon (see HorizonRot)
Return Type	Type (int), Description (string), Channel (int), InRange (bool), Relative Position (Vector)

NAV:<"FOCUS", vessel name or index>:NavaidRelVel

Reference API method	None. Returns velocity relative to navaid, in different reference frames, including vessel and orizon relative.
Additional Arguments	Receiver number
Return Type	Type (int), Description (string), Channel (int), InRange (bool), Relative Velocity (Vector)

Planetary Body Messages

These messages perform operations for Planetary Bodies (Sun, planets, moons). Their primary use entails getting information about bases and their navigational resources. These messages normally take an object identifier (name or index) before the command name. Reference API is OrbiterAPI.

BODY:<name or index>:Period

Reference API method	oapiGetPlanetPeriod
Additional Arguments	None
Return Type	double

BODY:<object name or index>:Obliquity

Reference API method	oapiGetPlanetObliquity
Additional Arguments	None
Return Type	double

BODY:<object name or index>:Theta

Reference API method	oapiGetPlanetTheta
Additional Arguments	None
Return Type	int

BODY:<object name or index>:ObliquityMatrix

Reference API method	oapiGetPlanetObliquityMatrix
Additional Arguments	None
Return Type	MATRIX3

BODY:<object name or index>:CurrRotation

Reference API method	oapiGetVPlanetCurrentRotation
Additional Arguments	None
Return Type	double

BODY:<object name or index>:HasAtm

Reference API method	oapiPlanetHasAtmosphere
Additional Arguments	None
Return Type	bool

BODY:<object name or index>:AtmConsts

Reference API method	oapiGetPlanetAtmConstants
Additional Arguments	None
Return Type	ATMCONST as csv

BODY:<object name or index>:AtmParams

Reference API method	oapiGetPlanetAtmParams
Additional Arguments	radius (double)
Return Type	ATMPARAM as csv

BODY:<object name or index>:JCoeffCount

Reference API method	oapiGetPlanetJCoeffCount
Additional Arguments	None
Return Type	int

BODY:<object name or index>:JCoeff

Reference API method	oapiGetPlanetJCoeff
Additional Arguments	coeff index (int)
Return Type	double

BODY:<object name or index>:JCoeffs

Reference API method	oapiGetPlanetJCoeffs
Additional Arguments	None
Return Type	csv of all JCoefficients by index

BODY:<object name or index>:BaseCount

Reference API method	oapiGetBaseCount
Additional Arguments	None
Return Type	int

BODY:<object name or index>:BaseName

Reference API method	oapiGetBaseName
Additional Arguments	baseId (name or base index)
Return Type	string

BODY:<object name or index>:BaseEquPos

Reference API method	oapiGetBaseEquPos
Additional Arguments	baseId (name or base index)
Return Type	longitude, latitude, radius (double,double,double)

BODY:<object name or index>:BasePadCount

Reference API method	oapiGetBasePadCount
Additional Arguments	baseId (name or base index)
Return Type	int

BODY:<object name or index>:BasePadEquPos

Reference API method	oapiGetBasePadEquPos
Additional Arguments	baseId (name or base index) padIndex (int)
Return Type	longitude, latitude, radius (double,double,double)

BODY:<object name or index>:BasePadStatus

Reference API method	oapiGetBasePadStatus
Additional Arguments	baseId (name or base index) padIndex (int)
Return Type	int

BODY:<object name or index>:NavData

Reference API method	oapiGetNavChannel oapiGetNavFreq oapiGetNavPos
Additional Arguments	baseId (name or base index) padIndex (int) vessel name or id (optional)
Return Type	int, double, vector, bool (in range if vessel supplied)

BODY:<object name or index>:NavChannel

Reference API method	oapiGetNavChannel
Additional Arguments	baseId (name or base index) padIndex (int)
Return Type	int

BODY:<object name or index>:NavFreq

Reference API method	oapiGetNavFreq
Additional Arguments	baseId (name or base index)
	padIndex (int)
Return Type	double

BODY:<object name or index>:NavPos

Reference API method	oapiGetNavPos
Additional Arguments	baseId (name or base index)
	padIndex (int)
Return Type	vector

General Object Messages

The following messages are for generic object information/handling. These commands normally take an *object* identifier (name or index) before the command name. Reference API is OrbiterAPI.

OBJ:Count

Reference API method	oapiGetObjectCount
Additional Arguments	None
Return Type	int

OBJ:<object index>:Name

Reference API method	oapiGetObjectName
Additional Arguments	None
Return Type	string

OBJ:<object name or index>:Type

Reference API method	oapiGetObjectType
Additional Arguments	None
Return Type	int

OBJ:<object name or index>:IsVessel

Reference API method	oapiIsVessel
Additional Arguments	None
Return Type	bool

OBJ:<object name or index>:Size

Reference API method	oapiGetSize
Additional Arguments	None
Return Type	double

OBJ:<object name or index>:Mass

Reference API method	oapiGetMass
Additional Arguments	None
Return Type	double

OBJ:<object name or index>:GlobalPos

Reference API method	oapiGetGlobalPos
Additional Arguments	None
Return Type	vector

OBJ:<object name or index>:GlobalVel

Reference API method	oapiGetGlobalVel
Additional Arguments	None
Return Type	vector

OBJ:<object name or index>:RelPos

Reference API method	oapiGetRelativePos
Additional Arguments	reference object id (name or object index)
Return Type	vector

OBJ:<object name or index>:RelVel

Reference API method	oapiGetRelativeVel
Additional Arguments	reference object id (name or object index)
Return Type	vector

OBJ:<object name or index>:ByType

Reference API method	none
Additional Arguments	Object type (int)
Return Type	comma separated objects of the specified type

OBJ:<object name or index>:AttachCamera

Reference API method	oapiAttachCamera
Additional Arguments	None
Return Type	"OK"

Error Codes

This section describes the meaning of any error messages that might be received from OrbConnect.

ERR00 Missing Message - Nothing was received.
ERR01 Incomplete Message - Not enough of the message was received to interpret it.
ERR02 Message Not Understood - The message could not be determined or a non-specific error occurred while parsing the message. Check your spelling and formatting.
ERR03 Missing Argument(s) - At least one argument was missing
ERR04 Invalid Argument - An argument was of the wrong type (letter instead of number), or value was out of range.
ERR05 Not a Vessel - Argument required to be a vessel
ERR06 XRVesselCtrl Interface not supported - The vessel cannot be accessed through XCTL message.
ERR07 XRVessel not identified - Could not get access to a XRVesselCtrl interface
ERR08 Command could not be executed by Orbiter.
ERR09 Bad Pointer - The program could not resolve a specified object. Check spelling, case or index.
ERR10 Invalid Object - The object was not of the required type. If the message was for a body (planet) or vessel the object found may not of that type. Also check spelling, case or index.
ERR11 Invalid Reference Object - The reference object specified for a relative position or velocity could not be determined.
ERR12 Planet has no Atmosphere. Returned when the planet has no atmosphere or static pressure is < 0.001kPa. Check first using HASATM and/or getAtmPressue.
ERR13 Vessel has no elements.
ERR14 Vessel has no nav receivers.
ERR15 No information available – The message was understood, but there was no matching data.
ERR98 Buffer Overflow - The argument received was too long to handle. Strings are limited to 250 characters. Numbers are limited to 8 characters.
ERR99 Fatal Error. Something Bad Happened.

Known Issues

Occasionally, Orbiter can ‘lock up’ or become unresponsive. A restart of Orbiter and any OrbConnect clients may become necessary.

Pausing Orbiter will not halt communication, but because of the way Orbiter works, messages processed during the pause are not consumed by Orbiter in an orderly fashion and may cause an unexpected state when unpaused. Workaround: Check that Orbiter is not paused, especially before sending ‘set’ or ‘toggle’ commands. If paused, your client could treat it as a ‘telemetry dropout’ with appropriate visual indications and disabling buttons, etc.

Licensing

Source Code

The entire project, including source code is available as a separate download from OrbitHanger.
http://www.orbithangar.com/