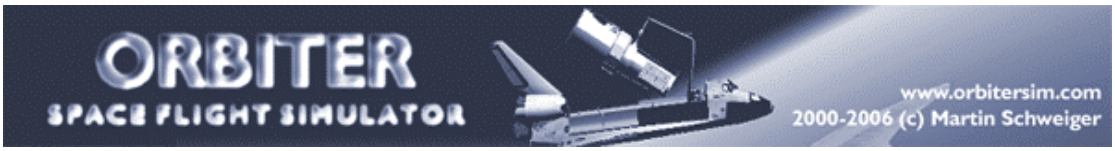
ORBITER Scenario Editor

 22 June 2006
Orbiter home: orbit.medphys.ucl.ac.uk/ or www.orbitersim.com



Contents

1 INTRODUCTION 2

2 USING THE EDITOR 3

2.1 Creating a new vessel 3
2.2 Configuring a vessel 3
2.3 Orbital elements 4
2.4 State vectors 5
2.5 Orientation 7
2.6 Angular velocity 7
2.7 Surface location 8
2.8 Propellant levels 9
2.9 Docking management 10
2.10 Deleting a vessel 11
2.11 Changing the simulation date 11
2.12 Saving a scenario 12

3 DEVELOPER SECTION 13

3.1 Vessel configuration files 13
3.2 Adding vessel-specific editor pages 13

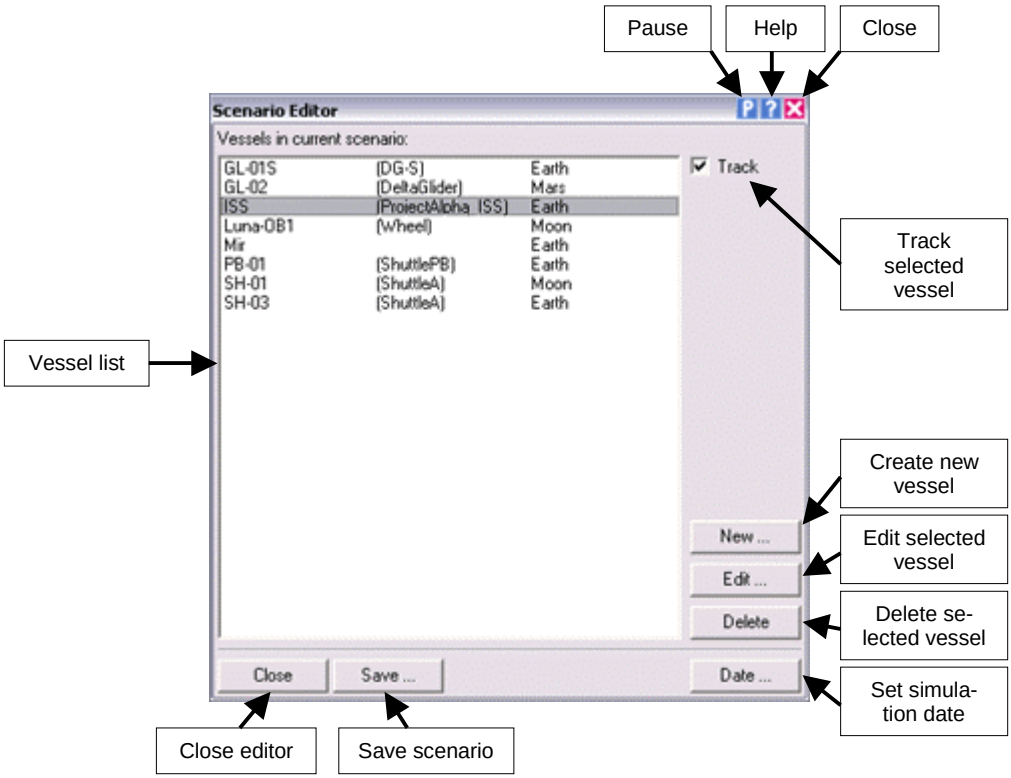
1 Introduction

Orbiter has an editor that allows to create, configure and delete vessels within a running simulation session.

You can set up the orbital elements, surface location, orientation, fuel states, docking connections, animation settings and other vessel-specific parameters for each vessel in the simulation. Once you are happy with the scenario, you can save it for later use, or share it with other Orbiter users.

The editor is provided as a plugin module. To use it, make sure that the *ScnEditor* module is activated in the *Modules* tab of the Orbiter Launchpad dialog (see Section “Modules Tab” in the Orbiter Manual on how to activate modules).

During the simulation, you can access the editor by opening the *Custom Functions* dialog with Ctrl F4, and double-clicking the *Scenario Editor* entry in the list. This will bring up the editor’s main page, containing a list of all vessels in the current simulation session.
Each entry contains the vessel name, class and the celestial body the vessel is currently orbiting or landed on. If you select a vessel from the list, Orbiter will automatically jump to that vessel, if the *Track* box is ticked.



You have now the option to:

- create a new vessel
- edit the vessel selected in the list,
- delete the selected vessel,
- change the simulation date, or
- save the scenario to a file.

2 Using the editor

2.1 Creating a new vessel

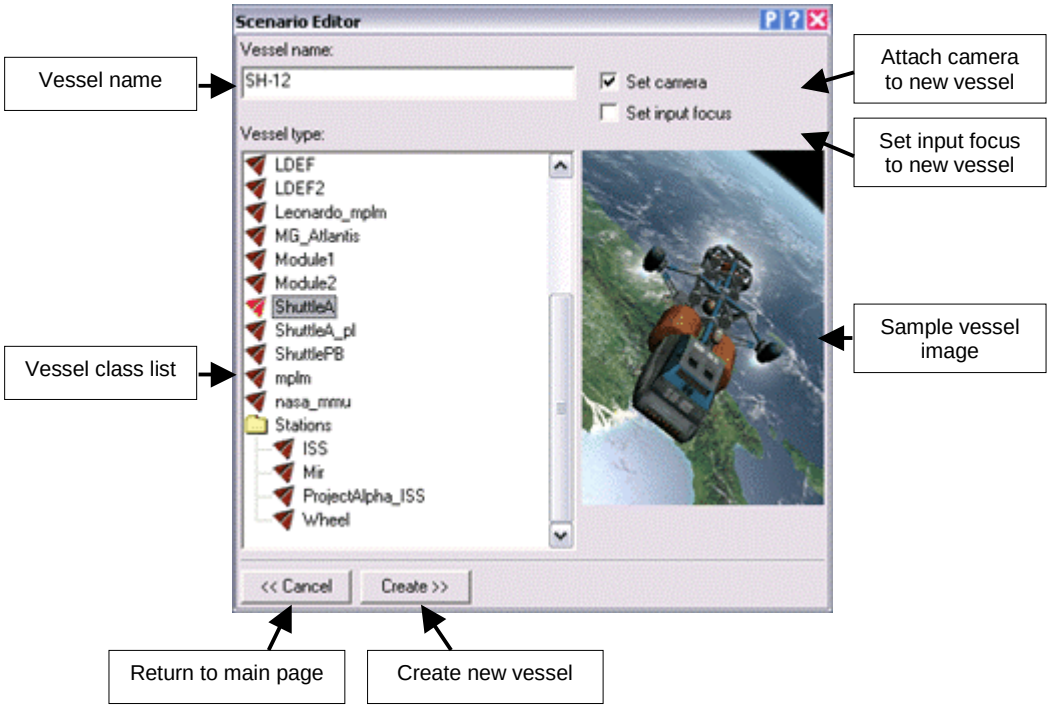
After clicking the *New* button on the main page, the editor will display the vessel creation page. It shows a list of all spacecraft types recognised by Orbiter. Note that some vessel types may be located in subfolders. If you want to see what a particular vessel type looks like, click on the entry in the list, and a picture is shown next to it. (Not all vessel types may support this feature.)

Select a type from the list, and enter a unique vessel name in the edit box above. If you enter a name that is already in use, or forget to select a vessel type, the editor will not allow you to proceed.

If you want the simulation to display the new vessel as soon as it is created, tick the *Set camera* box.

In addition, you can tick the *Set input focus* box to make the new vessel the focus object that receives user input via keyboard, mouse and joystick. Note that some vessel types may not accept input focus.

Click the *Create* button to generate the new vessel. All new vessels are placed in a default location (usually in a low Earth orbit).

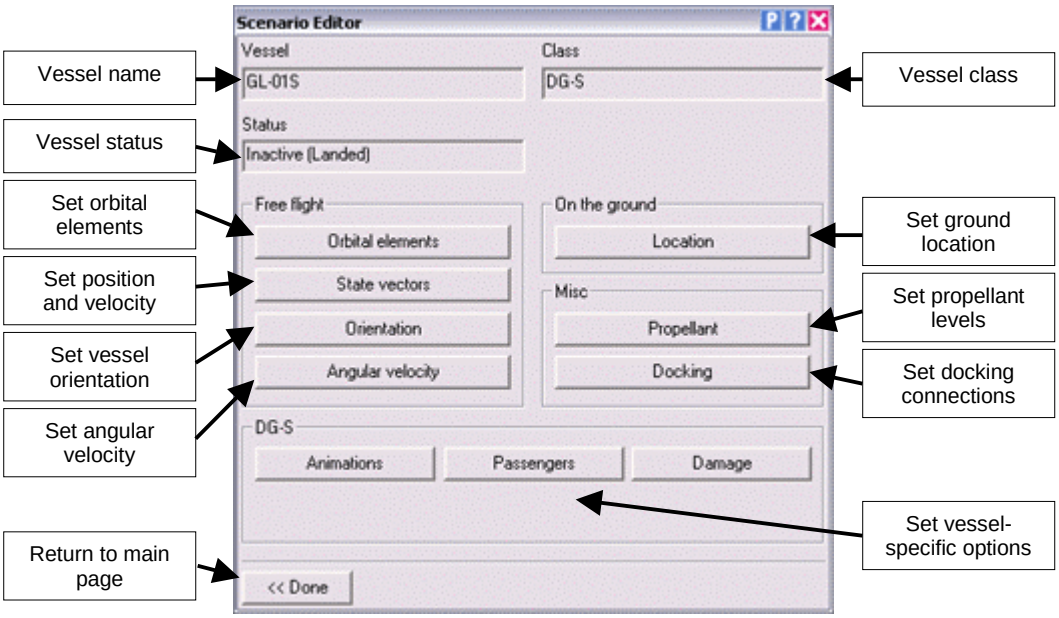


The editor now switches into *Configuration* mode. From here you can set the position, orientation and other state parameters of the new vessel.

2.2 Configuring a vessel

After clicking the *Edit* button on the main page, or after creating a new vessel, the editor will display the vessel configuration page for the selected vessel. From here you can control various aspects of the vessel's state, such as its position, velocity and orientation. You can also dock it to another vessel in orbit, and set vessel-specific parameters.

Note: Although not strictly necessary, it is sometimes useful to pause the simulation (by clicking the "P" button in the dialog title bar, or by pressing Ctrl P in the simulation window) before editing a vessel. Otherwise the continuously changing state parameters may make a precise configuration tricky.



2.3 Orbital elements

Use this page to place a vessel into orbit around a celestial body.

First select the object (planet, moon or sun) around which you want your vessel to orbit in the *Orbit reference* box.

Next, in the *Frame* box, select the coordinate frame in which the elements will be expressed. The choices are *ecliptic* and *ref. equator*. The ecliptic frame is defined by the plane of Earth's orbit, and the direction of the vernal equinox (at epoch J2000.0). This frame is useful e.g. if you want to place a vessel into an orbit for an interplanetary transfer. The equatorial frame is defined by the equatorial plane of the reference body. It is useful if you want to place the vessel into a specific orbit with respect to the planet surface, e.g. geostationary or polar.

Finally, set the *reference epoch* for the elements. This is the date to which the *mean longitude* parameter of the orbital elements will refer. If you select *current*, then the value will be interpreted as the mean longitude at the current simulation time. Otherwise, the value will represent the mean longitude at the specified *MJD* (Modified Julian Date). The latter option is more useful when editing a scenario without pausing, because a vessel's mean longitude is continuously changing along its orbit.

Now you are ready to set the elements in the left column of the dialog box. Below is a short description of the meaning of these parameters.

Semi-major axis [SMa]
For elliptical orbits, this defines the largest semi-diameter of the orbital trajectory. For hyperbolic orbits, SMa is negative and denotes the distance from the intersection of the hyperbola asymptotes to periapsis.

Eccentricity [Ecc]
Defines the shape of the orbit:

$e = 0$ circular orbit

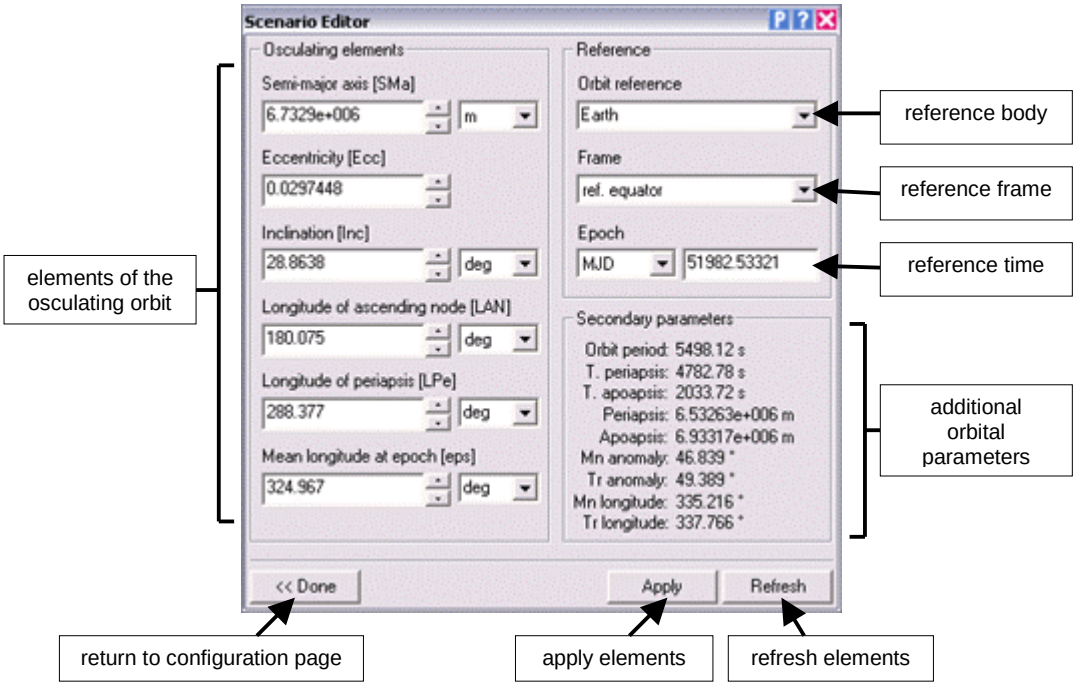
$0 < e < 1$	elliptic orbit
$e = 1$	parabolic orbit
$e > 1$	hyperbolic orbit

Inclination [Inc]
Defines the angle between the orbital plane and the reference plane. For example, if you selected "ref. equator" frame, then setting $i = 0$ will put the vessel into an orbit above the planet's equator, while $i = 90$ defines a polar orbit.

Longitude of ascending node [LAN]
The angular distance between the reference direction (e.g. vernal equinox) and the line of nodes (the intersection between the orbital and reference planes).

Longitude of periapsis [LPe]
The sum of LAN and the argument of periapsis (the angle between periapsis and the line of nodes).

Mean longitude at epoch [eps]
This defines the vessel's position along the orbital trajectory. The mean longitude is the sum of LAN and the mean anomaly (the angle between periapsis and the vessel position for a hypothetical circular orbit with the same orbital period as the true orbit). The value entered here refers to the specified epoch. For epoch = "current", the value represents the vessel's current (continuously varying) mean longitude. Otherwise, it is the longitude at the specified date (fixed for a given orbit.)



2.4 State vectors

Use this page to set the *position* and *velocity* of the vessel relative to a celestial body.

First, select the desired celestial body (sun, planet or moon) in the *Orbit reference* box. All position and velocity parameters will be interpreted relatively to the centre of this body.

Next, you should select the reference coordinate frame in which the state vectors are expressed. The *Frame* box provides three choices:

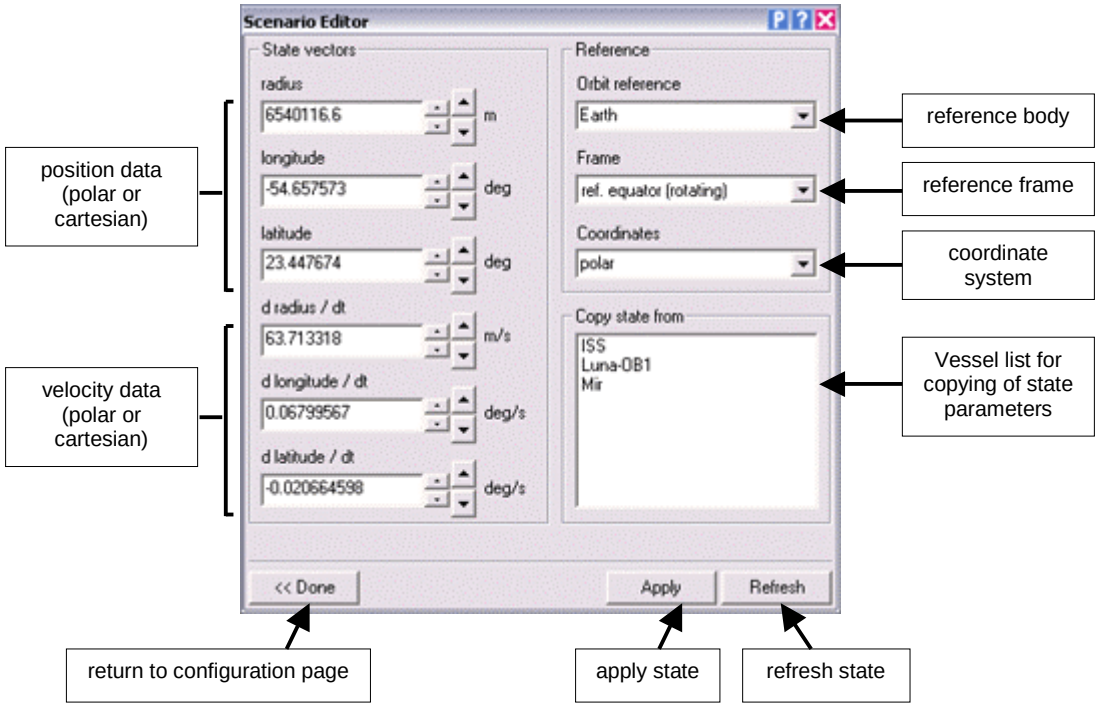
ecliptic: The ecliptic frame is defined by the plane of Earth's orbit around the sun. The x-axis is defined by the vernal equinox, the y-axis is the ecliptic north pole, and the z-axis is in the

plane of the ecliptic, perpendicular to x and y, and oriented so as to result in a left-handed coordinate system.

ref. equator (fixed): The coordinate system is defined by the equatorial plane of the reference body, with x and z in the equatorial plane, and y pointing to the north pole. The frame is non-rotating, i.e. fixed with respect to the celestial sphere.

ref. equator (rotating): As in the fixed case, this system is defined by the equator of the reference body, but it rotates with the planet, and the x-axis points to longitude 0°.

Finally, select the coordinates to use. You can select between cartesian coordinates (x , y, z and dx/dt, dy/dt, dz/dt) and spherical polar coordinates (r, φ, θ and dr/dt, dφ/dt, dθ/dt). Generally, polar coordinates will be more intuitive to use.



You are now ready to edit your position and velocity. Since a vessel's state vectors are constantly changing during the simulation, it is probably best to pause now by pressing the *P* button in the title bar. Then press *Refresh* to update the dialog data. Now you can enter new data for position and velocity, and press Apply to pass them to the simulation. Or use the spin controls next to the input boxes for direct manipulation.

Copy state

If you want to place your vessel next to another orbiting spacecraft, you can simply copy that vessel's state vectors over into the scenario editor, by clicking one of the entries in the *Copy state from* list. (Note that only vessels in flight are shown in the list.)

If you then click *Apply*, the copied state is applied to your vessel. Unless you had the simulation paused, the other vessel will probably have moved away in the meantime already. To copy and apply the state in a running simulation directly, you can double-click on the desired vessel in the list.

After copying your vessel onto the other spacecraft's position, you can edit the state vector entries to fine-tune the relative position of the two vessels.

Usage tips:

Editing the state vectors is probably most useful when your vessel is close to a planetary surface, for example during atmospheric flight. At orbital altitudes, it is easier to control the vessel state via its *orbital elements*. Defining a stable orbit via position and velocity data is much more difficult.

When editing a vessel close to the surface, the rotating equatorial frame is a good choice. In conjunction with polar coordinates, the position values translate directly to planetary longitude, latitude and radius data, and the velocity data are equivalent to ground speed in longitudinal, latitudinal and radial direction.

By editing its state vectors, a vessel is set to flight mode. Make sure the parameters you enter define a position above the surface. If you want to place a vessel on the ground, use the *Surface location* page instead.

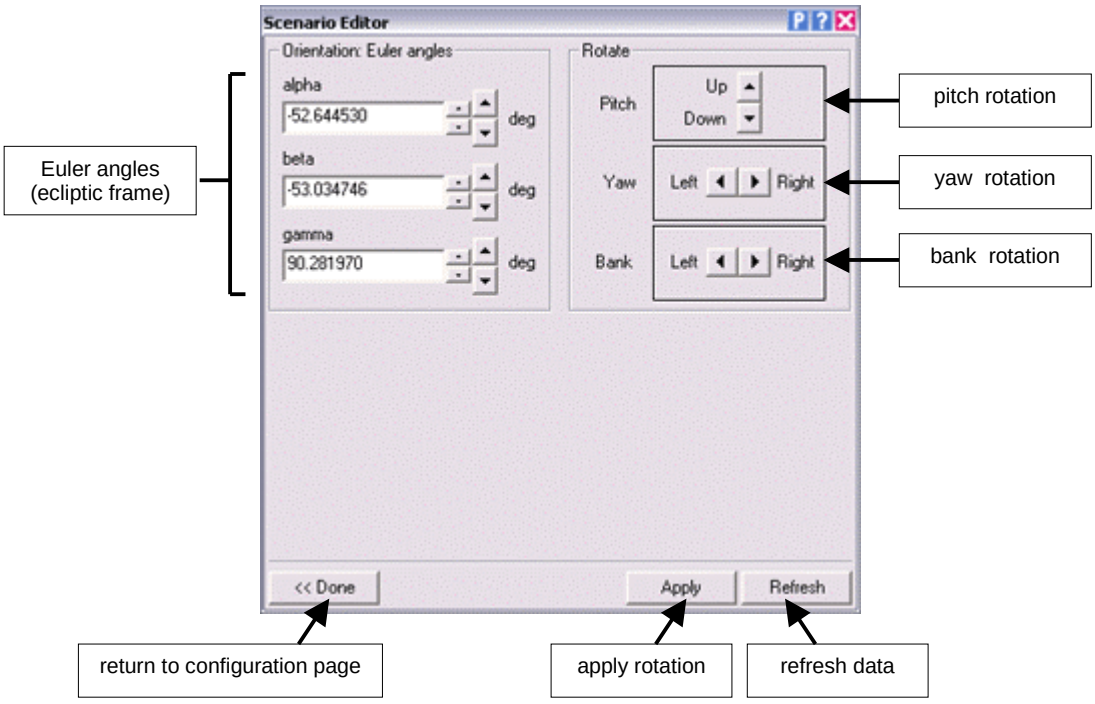
2.5 Orientation

This page allows to change the orientation of a vessel in free flight. You can either specify the *Euler angles* of the vessel frame with respect to the global ecliptic frame of reference, or you can *rotate* the vessel around its pitch, yaw and bank axes.

The Euler angles α, β and γ define the rotation matrix **R** which transforms from the vessel local frame to the global ecliptic frame as follows:

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

More intuitively, you can rotate the vessel around its local axes by pressing the pitch, yaw and bank buttons. The pitch buttons rotate around the vessel x-axis, the yaw buttons around the y-axis, and the bank buttons around the z-axis.



If the vessel is part of a composite structure (e.g. a spacecraft docked to a space station), the whole structure will be rotated.

The rotation settings have no effect for vessels on the ground.

2.6 Angular velocity

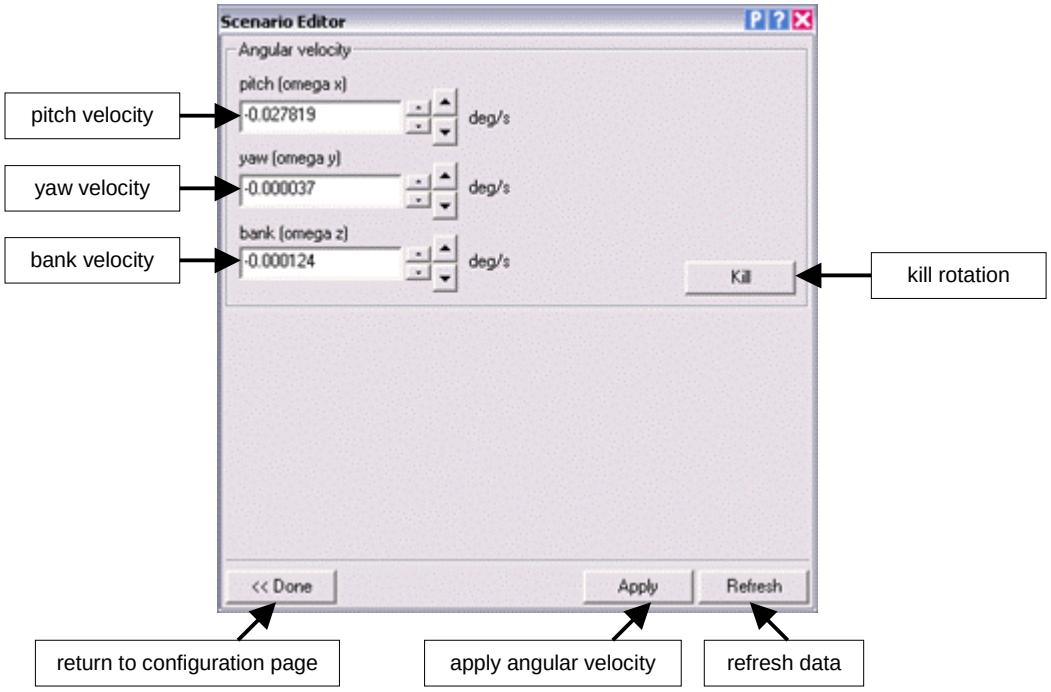
Use these controls to define the vessel's angular velocity around its three principal axes. You can adjust the velocity of rotation around the x-axis (pitch), y-axis (yaw) and z-axis (bank).

To terminate spin altogether, press the *Kill* button.

Note that the angular velocity components can oscillate between the three axes, depending on the vessel's principal moments of inertia (PMI). This means that pressing the *Refresh* button repeatedly may change the angular velocity values in the three axes, even if no additional torque is applied.

Likewise, pressing the *Apply* button repeatedly with the same angular velocity values can lead to abrupt changes in vessel spin.

The angular velocity settings have no effect for vessels on the ground.



2.7 Surface location

Applying options from this page will place your vessel on the surface of a planet or moon.

First you should select in the *Celestial body* box the body on which to place the vessel. If applicable, select a spaceport in the *Surface base* box, and a *Landing pad* for your vessel.

Alternatively, you can directly input the equatorial coordinates of your landing site in the *Longitude* and *Latitude* boxes.

Use the *Heading* box to adjust the orientation of your vessel on the ground.

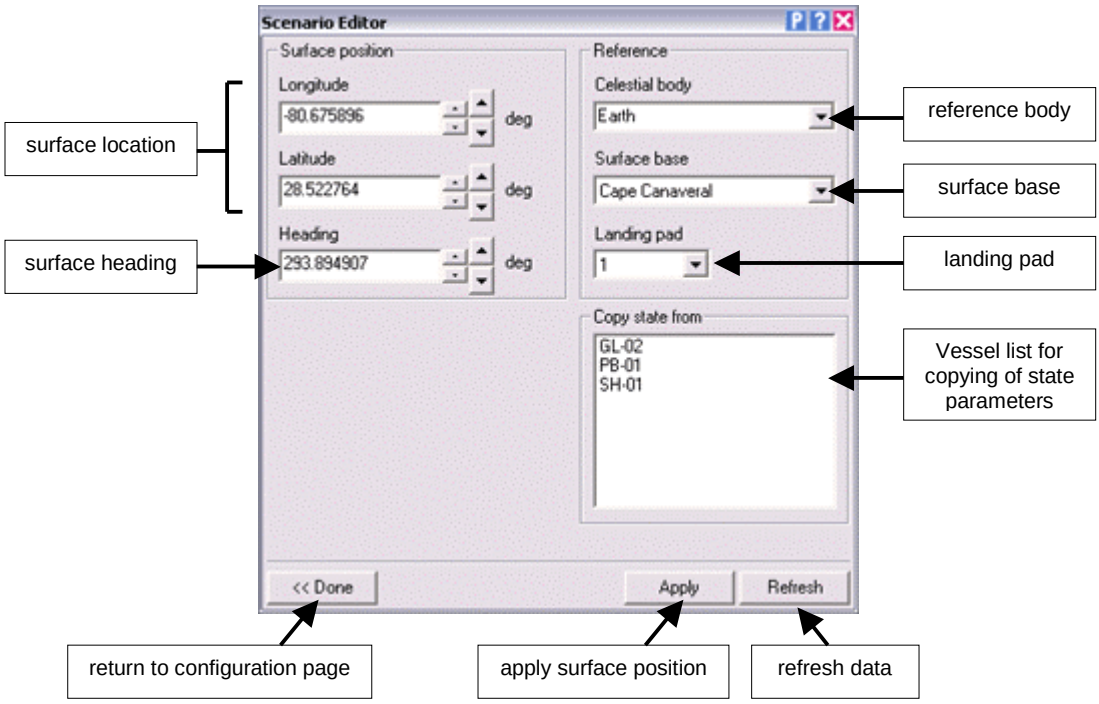
You can use the spin buttons next to the *Longitude*, *Latitude* and *Heading* boxes to adjust the values (small spinners scroll by small amounts, large spinners scroll by large amounts).

If you enter numerical values directly, you must press *Apply* to register the new values.

In the current version, docked assemblies of vessels on the ground are not supported. Therefore avoid assigning a surface position to a vessel that is part of a docked superstructure.

Copy state
Similar to the *Status vector* page, the *Surface location* page provides a list of vessels from which the surface parameters can be copied over. All vessels in the list are currently landed

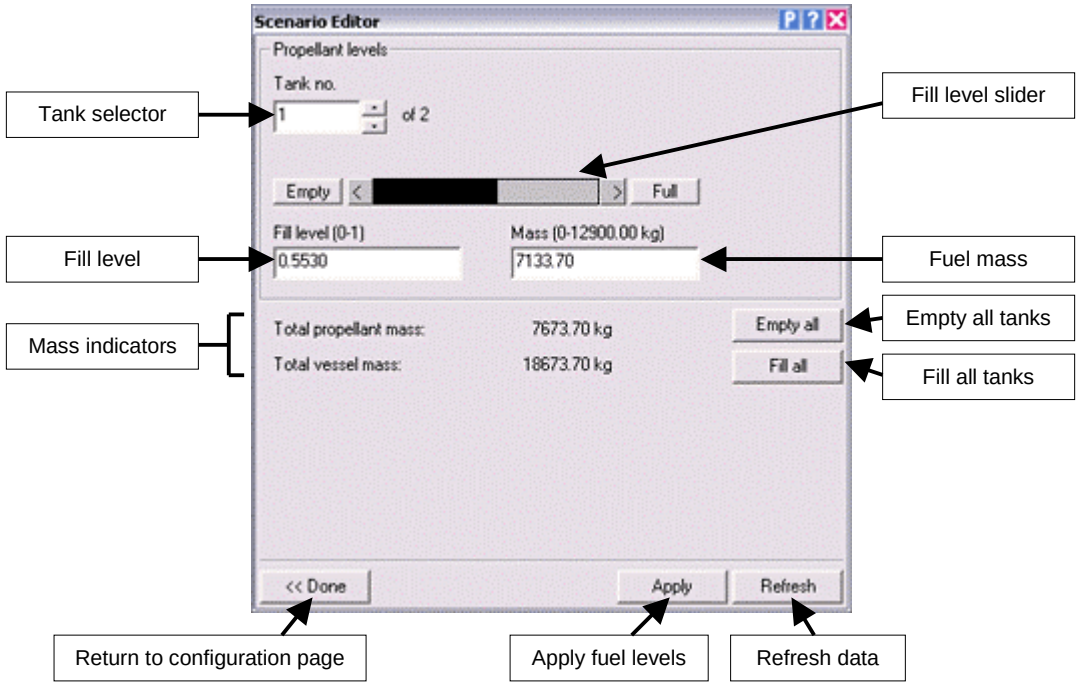
on a planetary surface. By clicking on an entry in the list, that vessel's surface parameters are copied into the scenario editor page. You can then click Apply to move your spacecraft to the specified vessel's location. Alternatively, double-clicking applies the copied parameters immediately.
Once you copied your spacecraft onto the other vessel, you can then edit the surface position parameters to fine-tune the relative position of the two vessels.



2.8 Propellant levels

On this page you can set the amount of propellant in each of the vessel's fuel tanks.

Select a tank in the *Tank no*. box. Then you can use the slider to define the propellant fill level in that tank. Alternatively, enter a fill level (0-1) or the fuel mass in one of the text boxes below, and press *Apply*.
To empty or fill all fuel tanks simultaneously, use the *Empty all* and *Fill all* buttons.



2.9 Docking management

If the vessel is equipped with docking ports, this page can be used to set up docking connections to other vessels, or to disengage existing connections.

First you need to select the docking port you want to edit. You can scan through the available ports with the *Dock no*. control in the top left. Many spacecraft have only a single docking port, but some – in particular space stations – may have several.

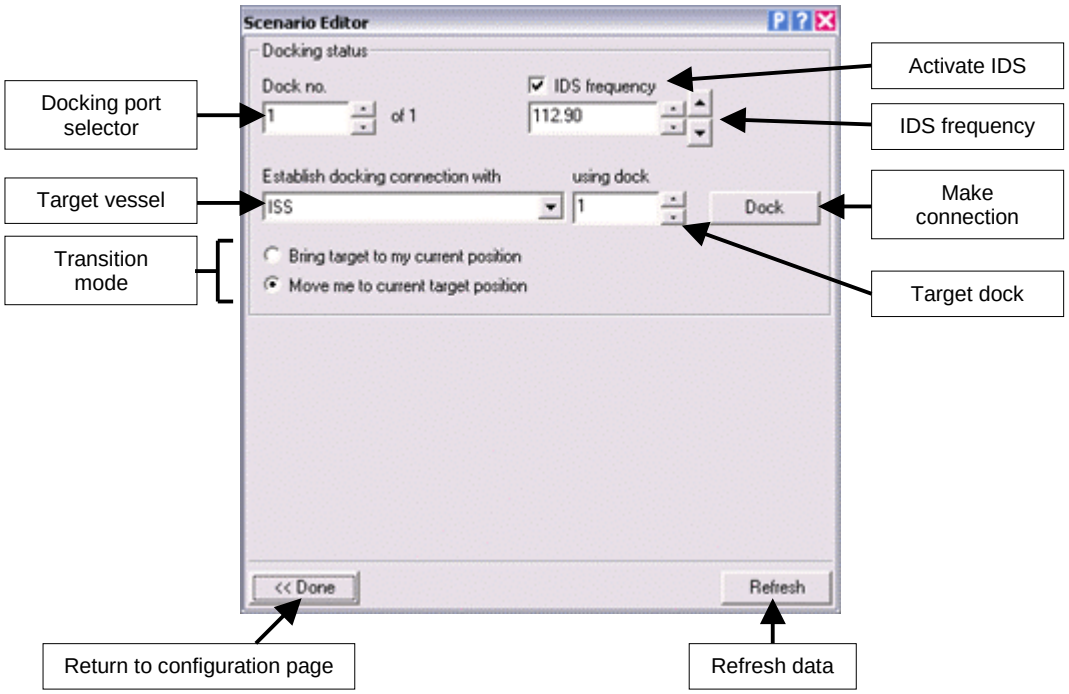
Next, you can enable or disable an IDS (Instrument Docking System) transmitter for that dock, using the controls in the top right of the page. IDS transmitters allow other vessels to use their Docking MFD and HUD modes for instrument-assisted docking approaches. If you enable the IDS, you can also set the transmitter frequency, from 108.00 to 139.95 MHz in 0.05 MHz steps. Make sure that each docking port of your vessel uses a different IDS frequency!

Finally, you can manage the docking connections at the port. If a vessel is already docked, its name will be displayed. You can click *Undock* to break the docking connection.

If the dock is not engaged, you can pick a vessel from the list provided, and click *Dock* to connect the vessel at the selected dock. If the target vessel has more than one docking port, you also need to chose the connecting port on the target vessel.

You can also select how the two vessels will be moved to make the connection. If you click *Bring target to my current position* then the vessel you are editing will maintain its position, and the target vessel will be dragged to the docking port on your vessel. If you click *Move me to current target position* then your vessel will be dragged to the target vessel instead.

Note that docking connections between landed vessels are not currently supported. Therefore, at least the vessel that maintains its position must be in flight. Docking two landed vessels, or dragging a flying vessel to the position of a landed one will result in undefined behaviour.



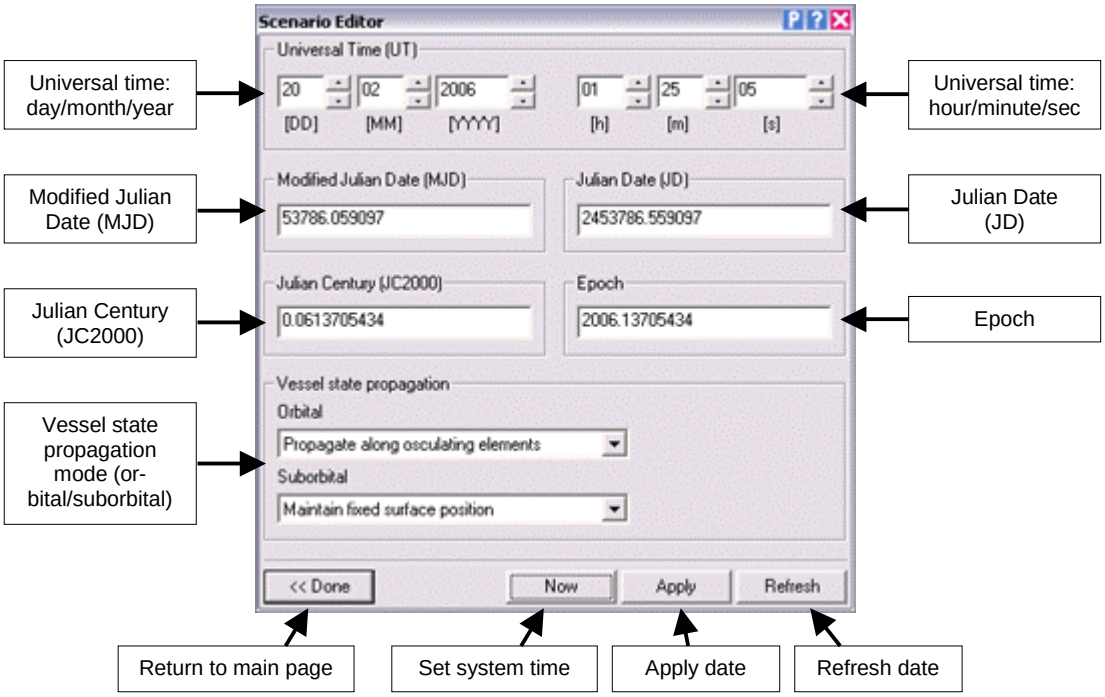
2.10 Deleting a vessel

Clicking the *Delete* button in the editor's main page will delete the vessel that is selected in the vessel list. The input focus and camera will automatically switch to another vessel, if necessary.

You are not allowed to delete all vessels from the simulation: Orbiter requires at least one vessel that accepts control input to work properly (note that some vessels may have been designed not to receive the input focus.)

2.11 Changing the simulation date

The *Date* button on the editor main page opens the date selection page. Here you can change the date of a running simulation, and specifiy how vessels will be propagated from the current to the new date.



The simulation date is displayed in different formats:

- **Universal Time (UT)**: A global reference time based on siderial time. Corresponds to the local time at longitude 0°.
- **Julian Date (JD)**: The interval of time in days and fractions of a day since 1 January 4713 BC, Greenwich noon.
- **Modified Julian Date (MJD)**: The Julian date minus 240 0000.5.
- **Julian Century (JC2000)**: The interval of time in centuries and fractions of a century since 1 January 2000, Greenwich midnight.
- **Epoch**: Year and fraction of a year.

If any of the date fields is modified, all other date formats are adjusted automatically. Clicking the *Refresh* button updates the dialog fields with the current simulation time. Clicking the *Apply* button sets a new simulation time, and clicking the *Now* button sets the simulation time to the current computer system time. By using the spin controls in the UT section, the simulation time is updated directly.

Before changing the simulation time, you should specify how vessels are propagated in time to the new date. Different time propagation modes can be selected for orbiting vessels (with orbits that don't intersect the central body) and for suborbital vessels (with trajectories that intersect the surface). The following options are available for orbital vessels:

- **Maintain fixed state vectors**: keep the vessel's relative position and velocity with respect to the central body fixed in a non-rotating frame. This means that the planet is rotating underneath, while the vessel keeps its location and attitude.
- **Maintain fixed surface position**: keep the vessel's position velocity and attitude fixed relative to the planet surface. This means that the vessel is rotating together with the planet, and stays fixed above the same point of the surface.
- **Propagate along osculating elements**: move the vessel along its current orbital trajectory, assuming that no forces other than the central body's gravitational force are acting on the vessel.

For suborbital vessels, in addition to the above one further option is available:

- **Destroy vessels**: destroy any suborbital vessels (i.e. assume that the vessels impacted on the ground during time propagation).

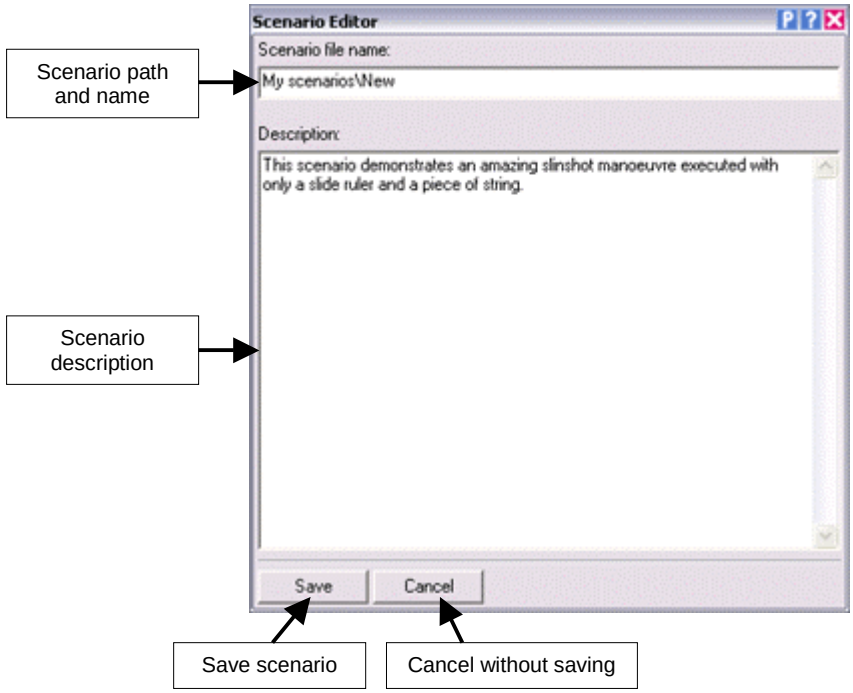
You can adjust the time forwards as well as backwards, but remember that moving back in time will not necessarily restore the simulation to a previous state, because the vessel propagation does not take into account events such as engine burns.

2.12 Saving a scenario

Once you are happy with the scenario you have created, you can save it to a scenario file which you can run at a later time or share with other Orbiter users.

To save the current simulation state, click the *Save* button on the main editor page. This will open the *Save scenario* page.

Enter a file name and a short scenario description. Optionally, the file name can include a directory path. All paths are relative to Orbiter's root scenario folder (*Orbiter\Scenarios*). The specified directory must already exist.



The text you enter in the description box will be displayed in Orbiter's Launchpad dialog when a user selects the scenario from the list. It should briefly explain what the scenario is about.

Finally, click *Save* to save the scenario to file, or *Cancel* to return without saving.

3 Developer section

This section contains information for vessel addon developers who want to prepare their vessel plugins to work with the scenario editor.

3.1 Vessel configuration files

The scenario editor can create instances of any addon vessels the user has installed in his or her Orbiter directory. The only requirement is that the addon vessel configuration (.cfg) file is placed in the *Config\Vessels* directory. Whenever the editor displays the *Vessel creation* page, it scans this directory and lists all vessel types for which configurations have been found.

Vessel configuration files can also be placed in subfolders below the *Config\Vessels* directory, for example *Config\Vessels\MyAddon\MyVessel.cfg*. The Vessel creation page will then show these subfolders in the list, and the user can descend into them. The path relative to the *Config\Vessels* folder is then part of the vessel class name, and is included in the vessel class specification in all scenario files referring to these vessel types.

Scenario example:

```
[...]
BEGIN_SHIPS
MyShip-01:MyAddon\MyVessel
  [...]
END
[...]
END_SHIPS
```

When editing scenario files by hand, remember to include the config file path. Otherwise Orbiter will not find the configuration files and terminate with an error message.

If you don't want a vessel type to appear in the vessel creation list, add the entry

```
EditorCreate = FALSE
```

to its config file. The scenario editor skips config files with this entry when populating the vessel type list.

Note: In older Orbiter versions, vessel configuration files were placed directly in the *Config* directory. Placing vessel configuration files in *Config* is still valid, but the editor will not recognise these vessel types in the creation page. Therefore, placing a vessel config file in the Config folder has the same effect as the "EditorCreate = FALSE" config tag.

3.2 Adding vessel-specific editor pages

Vessel addon developers can include editor pages in their vessel addon projects to allow users to configure vessel-specific options from within the Scenario editor. These may include animation settings, payload management, launch stack configuration, etc.

To configure editor pages for your vessel class, you need to compile a DLL which contains the dialog page templates and the code that responds to user input. If your vessel is already controlled by a DLL, you can add the editor elements to that. Alternatively, you can put the editor elements into a separate DLL. This is particularly useful if your vessel is managed by a script-driven wrapper DLL such as spacecraft.dll.

To tell Orbiter where to find the editor extensions, the vessel configuration (.cfg) file must contain the line

```
EditorModule = <editor-module>
```

where <*editor-module*> is the name of the DLL containing the editor elements. If you are using the vessel DLL for this, the `EditorModule` value is identical to the `Module` value. If the configuration file does not contain an `EditorModule` entry, the editor will still allow to configure the vessel's generic parameters, but won't provide any vessel-specific options.

The editor extension DLL must contain a *secInit* callback function (sec = scenario editor callback) with the following format:

```
DLLCLBK void secInit (HWND hEditor, OBJHANDLE hVessel)
```

where `hEditor` is the editor window handle, and `hVessel` is the vessel instance handle.

Vessel-specific options in the scenario editor are accessed by creating additional buttons in the *Vessel configuration* page (see Section 2.2). Up to 6 custom buttons can be added. You can create a button in the body of the *secInit* function in two ways:

Creating an embedded editor page
To create a button that opens a custom page inside the scenario editor dialog box, add code of the following form in the body of *secInit*:

```
EditorPageSpec eps = {
     "Animations", hDLL, IDD_EDITOR_PG, EdPgProc};
SendMessage (hEditor, WM_SCNEDITOR, SE_ADDPAGEBUTTON,
     (LPARAM)&eps);
```

EditorPageSpec is a structure that contains the parameters for the custom editor function. It has the format

```
typedef struct {
     char btnlabel[32];
     HINSTANCE hDLL;
     WORD ResId;
     DLGPROC TabProc;
} EditorPageSpec;
```

where

btnlabel	label displayed on the new button in the configuration page
hDLL	module instance handle
ResId	resource identifier for the embedded dialog page
TabProc	dialog page window procedure

ResId is the identifier of a dialog resource defined in the DLL. Its size and layout should conform to the standard dialog pages of the scenario editor. The simplest way to create a conforming dialog page is by copying one of the standard pages from the scenario editor resource file (*Orbitersdk**samples**ScnEditor**ScnEditor.rc*), and then modifying the page in a resource editor.

TabProc must be implemented in the DLL as a standard Windows dialog message procedure, i.e. with the interface

```
BOOL CALLBACK TabProc (HWND hWnd, UINT uMsg, WPARAM wParam,
     LPARAM lParam)
```

This function receives all user input to the custom page like an ordinary dialog message procedure. The module can use the messages sent to the message procedure to set or display vessel parameters. To obtain a handle to the vessel inside the message procedure, use

```
OBJHANDLE hVessel = (OBJHANDLE)lParam;
```

when processing the *WM_INITDIALOG* message, and

```
OBJHANDLE hVessel;
SendMessage (hDlg, WM_SCNEDITOR, SE_GETVESSEL,(LPARAM)&hVessel);
```

when processing any other messages.

Creating a callback function

Instead of creating an embedded dialog page, you can simply request the editor to notify your module whenever a custom button is pressed. This gives more freedom over the custom functionality you are adding. For example, you can open a separate dialog box, or perform some other action.

To create a button that calls a notification function in your module, add the following code in the body of *secInit*:

```
EditorFuncSpec efs = {
     "Payload", EditorFunc};
SendMessage (hEditor, WM_SCNEDITOR, SE_ADDFUNCBUTTON,
     (LPARAM)&efs);
```

EditorFuncSpec is a structure that contains the parameters for the custom button. It has the format

```
typedef struct {
     char btnlabel[32];
     CustomButtonFunc func;
} EditorFuncSpec;
```

where

btnlabel	label displayed on the new button in the configuration page
func	callback function in the module to be called after the button is pressed

The *CustomButtonFunc* must be defined in the module with the following format:

```
void EditorFunc (OBJHANDLE hVessel)
```

It receives the vessel handle as an input parameter.

An example for the implementation of vessel-specific editor pages can be found in the *Delta-Glider* vessel module that comes with the Orbiter SDK (*Orbitersdk\samples\DeltaGlider*).