



Prometheus Launch Vehicle
By Zatnikitelman

Contents

License	2
Vehicle Description	3
Technical Description	3
Flying Prometheus	4
Launchpad	4
Payload Management	5
Adding Payloads	6
Scenario File	6
Keys	8
Uninstalling	9
Troubleshooting	9
Changelog	10
Acknowledgements	10

License

Addon refers to the file in which you originally received the Prometheus addon in, and all its contents.

By using any part of this addon, you agree to the following terms.

The creator of the addon assumes no liability whatsoever for any result of your use of this addon.

You must not use any part of this addon for illegal or otherwise unauthorized activities.

The creator of this addon does not grant permission for redistribution except under explicit permission from the creator.

All parts of this addon are original work by the creator.

Any part appearing to be derivative or identical is purely coincidental, and/or part of a standard process (i.e. the structure of the vessel's code is similar to all other Orbiter vessels out of necessity for interfacing with Orbiter).

Should you not agree to the terms here, you must destroy any and all copies of this addon in your possession.

Vehicle Description

The Prometheus launcher for Orbiter is intended to represent a possible EELV-style heavy-lift vehicle using inspiration from the Atlas V, Delta IV, and Angara launchers. As such, Prometheus makes use of the “Common Core System” (CCS) where several identical modules are attached to each other to enable greater payload capacity. Whereas the other vehicles mentioned above apply this only to the first stage, Prometheus uses common second stage modules to provide greater lift capability.

Technical Description

Prometheus is an all cryogenic vehicle similar to the Delta IV using Liquid Hydrogen fuel and Liquid Oxygen oxidizer in all stages. Engine gimbaling is used during first stage flight to steer the vehicle. As previously described, Prometheus makes use of the CCS to provide greater payload capability at reduced cost versus using specialized staging. Each CCS is mostly identical to others of its stage (1st stage, or 2nd stage) and can be used in any position on that stage. To increase launch pad compatibility, the CCSs share fuel and oxidizer lines between parallel stages so launch pad plumbing is simplified in only having to interface with one CCS. The CCSs do however have to be slightly modified to be used as core stages of the vehicle. This simply adds the stage release latches and in the case of the core CCS2, resistojet thrusters and a Static Grapple Fixture.

The center second stage as mentioned is slightly modified from the CCS standard to include resistojet thrusters and a Static Grapple Fixture (SGF) to allow capture by a robotic arm. (The SGF is similar to and compatible with the STS/ISS FRGF and PDGF). The second stage is designed in this manner to serve as a tug stage to bring a payload to an on-orbit target.

The Prometheus family consists of four launchers. The 1 model uses a single CCS for both first and second stages and can launch 15 tonnes to Low Earth Orbit. The 2 model uses 3 CCSs (two attached to

the first stage) on both stages and can launch 60 tonnes to LEO. The 3 model has 3 CCSs arranged around the center CCS and can launch 90 tonnes to LEO. The 4 model has 4 CCSs arranged around the center and can launch 110 tonnes to LEO.

Prometheus includes 4 types of fairings in both 5m and 8m diameters to accommodate the widest range of payloads possible. In addition to the standard tapered fairing, a tubular version of each size fairing is available which can accommodate oversize payloads with their own atmospheric fairings, including human-crewed vehicles. Also as an option with the tubular fairing is the use of a Launch Escape System tower to safely evacuate human-crewed payloads in the event of an unanticipated disassembly of the launch vehicle or other critical problem.

General Technical Specs of Prometheus	
1 st Stage Sea Level Engine Thrust	6.0MN
1 st Stage Vacuum thrust	6.6MN
1 st Stage Exhaust Velocity	4,118 m/s
2 nd Stage Vacuum Engine Thrust	800KN
2 nd Stage Exhaust Velocity	4,412 m/s
5m Fairing Mass	5000 kg
8m Fairing Mass	8000 kg

Limitations

- Name of vessels used as payloads limited to 256 characters.

Flying Prometheus

Prometheus is currently designed to be hand-flown, thus the user is responsible for steering the vehicle. Prometheus uses a unique parallel-burn stage system so due to this complexity, includes an engine autothrottler activated and deactivated with the “U” key. Staging is also automatic. When the fuel in each “stage” is used up,

the stage will automatically jettison. It is recommended that guidance be assisted using LaunchMFD (not included). Prometheus should be manually flown until the roll to proper heading is achieved, then pitch “up” (nose comes down, “heads-down” flight profile) and begin following LaunchMFD guidance. Ideally, the launch profile should be something like this:

- Roll to your launch heading (top of ship points toward target heading)
- Once top of ship points to launch azimuth, begin following the LaunchMFD prompts to orbit.
- Once orbit is achieved, deactivate the autothrottler with the U key.

The caveat is that you should have LaunchMFD aim for a 130km orbit. Any altitude above this causes LaunchMFD to want to level the rocket's attitude rather than follow a dynamic pitch profile to orbit.

Launchpad

Prometheus includes a custom launchpad with custom effects. This launchpad can be spawned by including a tag within the scenario file definition. The launchpad consists of the tower gantry, to which two fueling arms and a crew-access arm with whiteroom are attached. The crew-access arm allows a great degree of flexibility in capsule height. The whole arm translates up and down between the gantry levels with the arm able to rotate a short distance vertically (which has the effect of moving the whiteroom up and down) to provide fine height adjustment to non-standard crewed-payload positioning.

Payload Management

Adding Payloads

The goal of Prometheus of course is to bring payloads to Orbit. Adding payloads is accomplished through the scenario file. A user will create the payload vessel and designate it as a Prometheus payload by using the PAYLOAD tag under the Prometheus scenario file definition. If following the name of the payload, nothing else is specified, Prometheus will automatically place the payload in the payload stack

using the Payloads' z-axis cross sections to determine spacing. This can be overridden by including the position of the payload after the name. Future versions of Prometheus will include the ability to change the payload's position while a scenario is running.

Scenario File

Prometheus uses several unique scenario file entries to define custom behavior of the vessel.

The following are scenario file entries that the user should make. Some, all, or none can be specified.

LES <int>

This will define a Launch Escape System that can be used to pull payloads away from the launch vehicle. The number after LES is the number of payloads the LES should take with it up on an emergency jettison. Currently, only a default Launch Escape Tower can be used, however this may be changed in the future. The LES will only be created if the fairing is of the tubular type (see FAIRING tag below). As the default fairing is not, the LES by default does not get created.

PAYLOAD <string> <double> <double> <double>

The string after this tag is the name of a payload in a scenario file that a user wants attached to Prometheus. This tag can be used more than once to specify multiple payloads. The vessel to be attached does not have to be completely defined as shown in the example below.

```
DG-01:Deltaglider
STATUS Landed Earth
END
```

Should Prometheus not be able to find a vessel with the name <string>, an entry will be made in Orbiter.log.

The doubles after the name are in case the user wants to specify

a position for the payload, and is used by Prometheus itself upon saving of the scenario.

When Prometheus attaches a payload that doesn't have its position pre-specified, it looks at the cross sections of the vessel and adjusts the location of the payload accordingly. This is meant to assist the user and by no means assures that the payloads will be aligned correctly.

FAIRING <int> <int>

This specifies the type of fairing to be used. The first int represents the size of the fairing (currently only 5 or 8) and the second int represents the fairing type, 1 for standard (tapered at the top), or 2 for tubular (open at the top). The default values are 5 and 1 respectively.

PADHEIGHT <float>

This specifies how high off the ground Prometheus should sit. The default value has the engine bells resting on the ground. This entry is not necessary if using Prometheus's included launchpad.

LAUNCHPAD <string>

This specifies that Prometheus should spawn a launchpad at the launcher's position.

The following are scenario file entries that Prometheus writes to the scenario upon a save and should not be edited by the user unless they are sure they know what they are doing.

CONFIG <int>

This specifies which configuration the launcher is currently in (0-3).

GUIDANCEACTIVE

This is a simple flag to tell Prometheus that the guidance is active when saved for example during a launch.

GUIDSTARTTIME <double>

This tells Prometheus when its automatic guidance was started.

Keys

The following is a list of keys and their functions in relation to Prometheus.

U	Engages the automated guidance (currently, only automatic engine-throttling is available)
J	Manually jettisons the next stage, or payload if there are no stages left.
E (pressed 3 times in 2 seconds)	Triggers the emergency jettison of the LES such that it carries with it however many payloads the user has told it to take.

CTRL+J	Jettisons the payloads at any time.
CTRL+F	Jettisons the fairing if not landed.
CTRL+E	Jettisons the LES without jettisoning any payloads (non-emergency)

Launchpad Keys

The following keylist is used when Prometheus's included launchpad has control focus.

Y	Swings fuel arms away from vehicle
U	Swings fuel arms toward vehicle
A	Translate crew arm assembly down
Q	Translate crew arm assembly up
D	Crew arm toward vehicle
S	Crew arm away from vehicle
P	Move the whiteroom down
L	Move the whiteroom up

Uninstalling

To uninstall Prometheus and all its files from your Orbiter install, simply run `uninstall.bat` found in the `ORBITER_ROOT\doc\PrometheusLV` directory, then manually delete the `ORBITER_ROOT\doc\PrometheusLV` directory if you wish. The uninstall script will only uninstall files included in the addon assuming you have not renamed them. In case you have created your own scenarios in the `ORBITER_ROOT\scenarios\PrometheusLV` directory, it will not be deleted, just the scenarios that were included in the addon.

Troubleshooting

Prometheus was designed to be as sturdy as possible, however, there is no guarantee that it will work 100%, 100% of the time. To help with troubleshooting, I have so far included 3 writes to `Orbiter.log` to assist the user in finding a problem. The log entries are specified as `Prometheus_<name>`: where `<name>` is the name of the ship in the scenario.

Vessel <code><vessel_name></code> was not found!	Prometheus could not find that name for a vessel.
--	---

For all other troubleshooting, you should try Prometheus in a clean Orbiter installation. If it doesn't work, then you can contact me (Zatnikitelman) on Orbiter-Forum via private message or on the #orbiter-forum IRC channel on irc.systemnet.info. If it does, then it is recommended you bring the clean Orbiter install to the state you have it in on the problem system one addon at a time. Support for addons is usually available either through their Authors as specified in their documentation, or via Orbiter-Forum. **I however claim NO responsibility for solving problems you may have.**

Changelog

Version 1.6.1 – First release for Orbiter10

Version 1.6 – First Public Release

Version 1.5 – Added support for a whole Prometheus “family”

Version 1.2 – Now includes launch pad

Version 1.0 – Initial build of Prometheus

Roadmap

Add Autopilot

Add in-scenario payload-management

Add multiple payload fairing lengths

Acknowledgements

This addon could not have happened without many people assisting me along the way. Below is a list in no specific order.

- Hielor of the Orbiter community for explaining jettison coordinates and velocity and some help with Keycomm.
- The team who originally created the idea of Keycomm and explained it on the Orbiterwiki.
- Woo482 and Kaito for Beta Testing Prometheus
- Tblaxland of the Orbiter community for explaining double pointers with dynamic memory allocation.

- Douglas Beachy of the Orbiter community for providing lots of C++ help over the years
- Notebook and jedidia of the Orbiter community for assisting in calculating fuel mass ratios of fuel tanks.
- Face of the Orbiter community for assisting me with the (as yet not-implemented) guidance system.
- Gary Williams for inspiring me to use on-screen annotations instead of the debug string.
- The people on the #orbiter, #orbiter-forum, #orbiterradio, and #nyx IRC channels for assisting and inspiring me in many small ways.
- The Orbiter community in general which has likely assisted me in other non-specific ways without which this addon would be impossible.
- Dr. Martin Schweiger for creating the incredible Orbiter Spaceflight Simulator for which this addon is created.