Continuity
User Guide

Walthzer

13th June 2022

Contents

Chapter 1

Getting Started

Continuity is a mission-persistance system used to **save and load** the mission state to the server in a multiplayer environment. The primary usecase is to create persistant Zeus missions.

1.1 Enabeling Continuity

To reduce server load continuity isn't enabled by default, but on a per PBO basis. In the Eden editor simple do:

Options ⇒ WFAR Continuity ⇒ Enable Continuity

With that continuity is now enabled for this PBO!

1.2 Settings

Besides the per-PBO flag to enable continuity there are also additional CBA settings you can configure, see Figure 1.1. The autoSaveInterval enables or disables autosaving and sets the interval between saves in minutes, any value above 0 enables autosaves. The concurrentAutoSaveLimit sets how many autosaves will be made before overwritting a previous autosave.

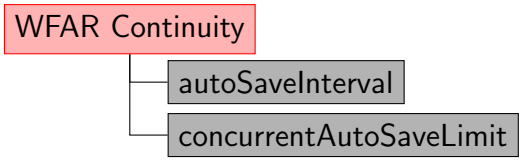

Figure 1.1: Continuity CBA Settings

1.3 Basics

To interact with continuity during a mission you must either: be an **Admin** or a **Zeus**.

Before the Mission

When loading into a mission you'll see a new button on the post-lobby map screen labeld: **Continuity Menu**. From here you can load any previous saves made for this PBO.

It's highly advised to only load a mission once and from this menu. Restart the mission before loading it again to avoid *unfortunate* results.

During the Mission

Whilst in a mission you can interact with continuity using **Chat commands**.

#continuity	Command	Parameter	
#continuity	load		**Open the load menu**
#continuity	save	saveName	**Save the mission**

When saving the mission if the savename is omitted continuity will create a QuickSave. Any admin or Zeus will be informed of newly created saves with a notification.

1.4 Save Structure

Before we can confidently start using continuity however, we must first understand a little about how it works. Continuity stores mission saveData under a combination of the PBO name and the time that PBO was saved in Eden.

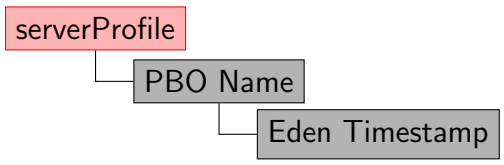

Figure 1.2: Continuity Save Structure

Suffix

Continuity considers the following PBO's to have the same name:

`bestMissionEver.pbo`	`bestMissionEver{NotReally}.pbo`

Any saves made on the former can be accesed and loaded on the latter and vice-versa. This `{NotReally}` is what we call the suffix. You can use it to label e.g. the parts in a multi-part persistant campaign.

Example

So if you save mission from Eden as **bestMissionEver.pbo** on the 11th of June and create a QuickSave on it. Then go back to Eden and add a few flags on the 15th of June. The load menu will resemble Figure 1.3. Continuity divides saves

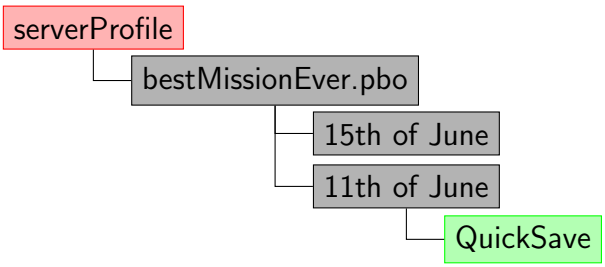

Figure 1.3: Example Save Structure

based on the Eden Timestamp to alert and prevent loading of incompatible and outdated saves. While still allowing you to **add** to PBO's in the Eden editor and load older saves.

Continuity will load saves for PBO's that have had objects removed. However to reduced the chance of accidents, consider using a suffix.

Chapter 2

Saved Information

With the basics down, we can now take a detailed look at what Continuity saves and how this is done.

2.1 saveData

Continuity collects all the mission data into a saveData container. Its a simple `key:value` hashmap. It stores some general mission data directly, but the actual meat of the save is subdivided in **saveList**s. See Figure 2.1 for a visual.

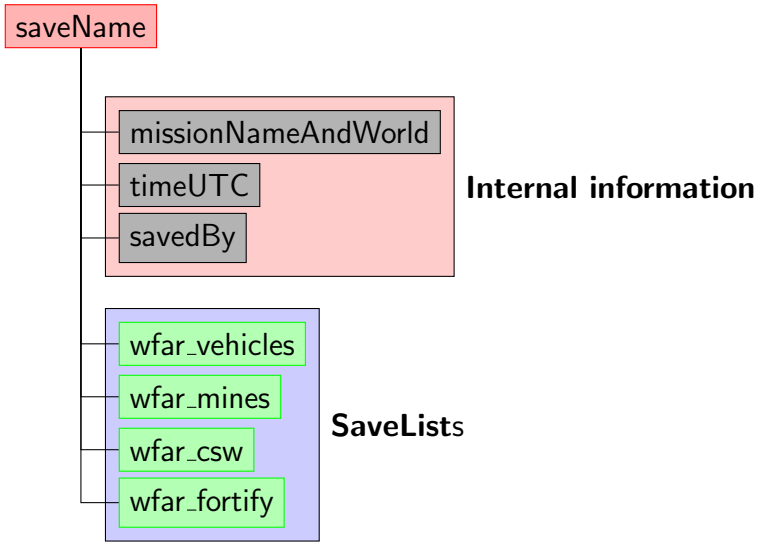

Figure 2.1: saveData structure

2.2 CfgWFARContinuity

This is the config class for Continuity, its primary use is to define the **saveList**s Continuity has and the respective save and load functions to use with those saveLists.

```
class CfgWFARContinuity {
    class saveLists {
        class wfar_vehicles {
            load="fnc_loadVehicles";
            save="fnc_saveVehicles";
        };
    };
};
```

2.3 saveList

The **saveList** itself is nothing more than an array Continuity either gets from a saveList's save function or uses as the argument for a load function. But by writting the associated function so that a saveList is self-contained and has no references to other saveLists, we in effect get a **mission save layer** that could be added or removed from a save at the user's discretion.

Continuity currently has no automated layer functionality but **manual** removal of saveLists from saveData is possible. Ask Walthzer if needed!

Functions

the save/load functions in Continuity differ widely depending on their saveList, but a simple example is wfar_mines

```
private _minesSaveList = [];
{
_minesSaveList pushBack [typeOf _x, getPosASL _x,
↪  [vectorDir _x, vectorUp _x]];
} forEach allMines;

_minesSaveList
```

```
params["_listSaveData"];
{
    _x params ["_type", "_posASL", "_vectorDirAndUp"];

    private _object = _type createVehicle _posASL;
    _object setPosASL _posASL;
    _object setVectorDirAndUp _vectorDirAndUp;

} foreach _listSaveData;
```

As you can see Continuity doesn't at all have much to do to save a mine. It simple stores the type of mine, its position and its orientation in the saveList. Then when loading the list, just creates a mine of the saved type at the saved position and sets the orientation! Ofcourse this is the simplest saveList most others are more involved, but there isn't a lower limit for how **basic** a saveList can be.

Enquire Walthzer about adding saveLists if you feel Continuity is lacking