Convert To String

Transform different data types into an executable string; by Faguss (ofp-faguss.com)

1. Overview

This function converts input value into a callable string. The purpose is to provide state of the variables in a text format that could be saved and loaded later.

2. Usage

- Copy `Convert_To_String.sqf` to the mission directory.
- Write in the mission `Init.sqs`:

```
CONVERT_TO_STRING = preProcessFile "Convert_To_String.sqf"
```

Function call syntax:

```
<data> call CONVERT_TO_STRING
```

Where `<data>` could be any of the following:

- String
- Number
- Boolean
- Side
- Array

Strings are wrapped in brackets. Array items are saved as a join formula in order to avoid single allocation limit.

Example:

```
[ "a", 1, [2] ] call CONVERT_TO_STRING
```

Output:

```
[] + [{a}] + [1] + [[]+[2]]
```

3. Converting Objects and Groups

a) Convert_Obj_To_String.sqf

Default version does not save objects and groups in a callable way. In OFP there's no good way to detect these types. One option is to use `"All" countType [_x]>0` but it has drawbacks:

- not every object inherits `class All` (e.g. island objects do not)
- it does not differentiate between objects and groups
- `objNull` and `grpNull` do not pass the condition

Even if you do use it there's no good way to turn back string into an object or a group type. One option is to use `nearestObject` which will work as long as the object has not moved.

This variant uses both of these methods. Example usage:

```
CONVERT_OBJ_TO_STRING = preProcessFile "Convert_Obj_To_String.sqf"
player call CONVERT_OBJ_TO_STRING
```

b) Convert_ObjGrp_To_String.sqf

More accurate way of detecting objects and groups is to eliminate all other possible data types. **Fwatch** is used in this version to find whether given value is a number or an object and as a result objects, groups, `objNull` and `grpNull` are converted properly.

```
CONVERT_OBJGRP_TO_STRING = preProcessFile "Convert_ObjGrp_To_String.sqf"
[player, group player, objNull, grpNull] call CONVERT_OBJGRP_TO_STRING
```

c) Convert_NamedObj_To_String.sqf

More reliable way of retrieving objects is to name them in the Mission Editor and then call these names. Downside is that all the units have to be manually named and the list of names has to be written down. Also the function won't be universal but mission-specific.

This package contains a script (in the demo mission **Automate_Naming.noe**) that uses Fwatch 1.16 to automate this process.

This variant checks if input value matches any of the named objects (or groups they are in). Multiple files are required:

- `Convert_NamedObj_To_String_Init.sqf` – prepares a list objects and groups (it has to be refreshed because units can respawn and/or they can change groups)
- `Convert_NamedObj_To_String.sqf` – converter function
- `FindInArray_OFPCWA.sqf` – search function

To use it call the init function and pass two arguments: data for the conversion and a list of names.

```
CONVERT_NAMEDOBJ_TO_STRING_INIT = preProcessFile "Convert_NamedObj_To_String_Init.sqf"
CONVERT_NAMEDOBJ_TO_STRING      = preProcessFile "Convert_NamedObj_To_String.sqf"
FindInArray_OFPCWA              = preProcessFile "FindInArray_OFPCWA.sqf"
IS_CWA = Format ["%1",getworld]!="scalar bool array string 0xfcffffef"

[[soldier1], ["soldier1"]] call CONVERT_NAMEDOBJ_TO_STRING_INIT
```

d) **Convert_AllObj_To_String.sqf**

This variant combines both the use of **Fwatch** to detect objects and the search for a named object.

```
CONVERT_ALLOBJ_TO_STRING_INIT = preProcessFile "Convert_AllObj_To_String_Init.sqf"
CONVERT_ALLOBJ_TO_STRING      = preProcessFile "Convert_AllObj_To_String.sqf"
FindInArray_OFPCWA            = preProcessFile "FindInArray_OFPCWA.sqf"
IS_CWA = Format ["%1",getworld]!="scalar bool array string 0xfcffffef"

[[soldier1, (object 61291)], ["soldier1"]] call CONVERT_ALLOBJ_TO_STRING_INIT
```

Conclusion: because OFP lacks a universal way to perform this conversion I recommend to write your own version of the function that will best suit your use case. Keep in mind that calling Fwatch is expensive (in terms of time) so use it only when you need it.

4. Version history

1.0 (09.07.21)
First release.