Working Title GNS Developer Notes

The Working Title GNS 430 and 530 bring a new level of authenticity to these units in Microsoft Flight Simulator, but due to the inherent complexity of their operations they may pose a number of challenges that aircraft developers may have to adapt to for full integration.

These notes are current as of GNS version: 1.0.1.

New Autopilot Options

The updated GNS 430/530 package allows for some autopilot integration options to be set.

- By default, the autopilot will support NAV mode arming for VOR and GPSS.
- By default, the autopilot will *not* support an independent flight director.
- By default, the autopilot will support Backcourse/REV mode.

To change any of these options, include any of these tags in your panel.xml file:

```
<Instrument>
	<Name>AS530</Name>
	<DisableAPNavArm>True</DisableAPNavArm>
	<SupportAPFlightDirector>True</SupportAPFlightDirector>
	<DisableAPBackCourse>True</DisableAPBackCourse>
</Instrument>
```

CDI/HSI Configuration Options

With the release of the updated GNS 430/530 new options for driving a CDI or HSI have been introduced. They now specifically support:

- Separate CDI source selection when multiple GNS units are installed;
- Driving vertical guidance from a GPS source (for RNAV-type approaches with vertical guidance);
- Support for dual GNS units of the same type (for example two GNS530s or two GNS430s);
- Support for hardware keyboard input to the GNS for selecting waypoints.

Legacy Mode/Backwards Compatibility

A cockpit panel developer may elect to not use these new capabilities at all, and the existing behavior will continue to function.

New CDI Source Selection Behavior

The updated GNS 430/530 package allows for each installed GNS unit to have its own CDI source selected, so for example you could have your GNS tied to COM/NAV1 set to GPS as the CDI source, while having a second GNS tied to COM/NAV2 set to VLOC as the CDI source, and the CDI “wired” to each GNS display correct annunciations and lateral/vertical guidance.

To use this new capability, developers must first update the panel.xml file to activate this new feature. To do this, find each `<Instrument>` tag for a GNS, and add a `<NewCDIBehavior>` tag under the Instrument, for example:

```
<Instrument>
	<Name>AS530</Name>
```

```
        <NewCDIBehavior>True</NewCDIBehavior>
        <NavIndex>1</NavIndex>
        <ComIndex>1</ComIndex>
</Instrument>

<Instrument>
        <Name>AS430</Name>
        <NewCDIBehavior>True</NewCDIBehavior>
        <NavIndex>2</NavIndex>
        <ComIndex>2</ComIndex>
</Instrument>
```

If the `<NewCDIBehavior>` tag is either omitted or set to false, legacy behavior will govern CDI behavior, and if the tag exists and is set to true, then the New CDI Behavior will govern.

Configuring the CDI for new behavior

Once the GNS has been configured for New CDI Behavior, you must also configure the actual CDI to listen to the correct vars from the simulator.

Linking the CDI Instrument to the GPS Instrument

The GNS units will index themselves, starting at index 1, for each type of GNS, and will publish CDI Source information in an LVAR according to the type of GNS and the index of that type of GNS. Examples:

- Aircraft with one GNS 530 as NAVCOM1, and one GNS 430 as NAVCOM2 (like the default steam gauge C172):
    - NAVCOM 1 is the first GNS530, so it will use `AS530_CDI_Source_1`
    - NAVCOM 2 is the first GNS430, so it will use `AS430_CDI_Source_1`
- Aircraft with two GNS 530s, one each as NAVCOM1 and NAVCOM2:
    - NAVCOM 1 is the first GNS530, so it will use `AS530_CDI_Source_1`
    - NAVCOM 2 is the second GNS530 also, so it will use `AS530_CDI_Source_2`

Setting CDI Source

The GNS unit will set the CDI Source LVAR (e.g. `AS530_CDI_Source_1`*)* to true (1) if the source for that CDI is GPS, and will set it to false (0) if the source is the tuned NAV radio (VLOC mode). In this way, the two GNS units can be set separately – for example one GPS unit can be set to GPS mode while the other is set to VLOC mode and the CDIs will display correctly for both radios.

Lateral Guidance Needle

The CDI should be configured to use the values from `A:GPS CDI NEEDLE` or `A:NAV CDI:`*`index`* depending on whether the source is set to GPS or VLOC.

RPN Example for a GNS530 configured as NAVCOM1

```
(L:AS530_CDI_Source_1, Bool) if {
        (A:GPS CDI NEEDLE, Number) 127 +
} els{
        (A:NAV CDI:1, Number) 127 +
}
```

Localizer (lateral guidance) Validity Flag

To detect whether there is a valid lateral guidance source for the CDI, you need to determine first what CDI source is set for the CDI and then determine whether there is valid guidance for that source.

RPN Example for a GNS530 configured as NAVCOM1

```
(A:GPS IS ACTIVE FLIGHT PLAN, Bool)
(A:NAV HAS NAV:1)
(L:AS530_CDI_Source_1, Bool) ?
```

Vertical Guidance Needle

The CDI should be configured to use the values from `A:GPS GSI NEEDLE` or `A:NAV GSI:`*`index`* depending on whether the source is set to GPS or VLOC.

RPN Example for a GNS530 configured as NAVCOM1

```
(L:AS530_CDI_Source_1, Bool) if {
        (A:GPS GSI NEEDLE, Number) 127 +
} els{
        (A:NAV GSI:1, Number) 127 +
}
```

Glideslope (vertical guidance) Validity Flag

To detect whether there is a valid vertical guidance source for the CDI GSI, you need to determine first what CDI source is set for the CDI and then determine whether there is valid guidance for that source.

RPN Example for a GNS530 configured as NAVCOM1

```
(A:GPS HAS GLIDEPATH, Bool)
(A:NAV HAS GLIDE SLOPE:1)
(L:AS530_CDI_Source_1, Bool) ?
```

Display GPS Vertical Guidance for RNAV-type approaches without using Split CDI Behavior

Developers can also choose to display vertical guidance for GPS approaches without having to implement the split CDI behavior if preferred. To do so, the following examples apply:

Vertical Guidance Needle

The CDI should be configured to use the values from `A:GPS GSI NEEDLE` or `A:NAV GSI:1` depending on whether the source is set to GPS or VLOC

RPN Example for a GNS unit configured as NAVCOM1:

```
(A:GPS DRIVES NAV1, Bool) if {
        (A:GPS GSI NEEDLE, Number) 127 +
} els{
        (A:NAV GSI:1, Number) 127 +
}
```

Glideslope (vertical guidance) Validity Flag

To detect whether there is a valid vertical guidance source for the CDI GSI, you need to determine first what CDI source is set for the CDI and then determine whether there is valid guidance for that source.

RPN Example for a GNS unit configured as NAVCOM1:

```
(A:GPS HAS GLIDEPATH, Bool)
(A:NAV HAS GLIDE SLOPE:1)
(A:GPS DRIVES NAV1, Bool) ?
```

Implementing dual GNS units of the same type (2 x GNS530, 2 x GNS430)

Developers can also choose to equip aircraft with two GNS units of the same type, for example two GNS 430s. In this scenario, the instruments need to be indexed so that the instance of the GNS loaded by the panel.cfg file can be matched up with the proper configuration options selected in the panel.xml file. To do this, there are three steps:

1) panel.cfg file needs to include an index that matches the NAVCOM index for the instrument. Example:

```
[VCockpitXX]
size_mm                 = 350,190
pixel_size             = 350,190
texture                 = $GNSScreen_1
htmlgauge00=NavSystems/GPS/AS430/AS430.html?Index=1, 0, 0, 350,190

[VCockpitXX]
size_mm                 = 350,190
pixel_size             = 350,190
texture                 = $GNSScreen_2
htmlgauge00=NavSystems/GPS/AS430/AS430.html?Index=2, 0, 0, 350,190
```

2) panel.xml file needs to reflect the instrument names with the same index:

```
 <Instrument>
       <Name>AS430_1</Name>
       <NewCDIBehavior>True</NewCDIBehavior>
       <NavIndex>1</NavIndex>
       <ComIndex>1</ComIndex>
</Instrument>

 <Instrument>
       <Name>AS430_2</Name>
       <NewCDIBehavior>True</NewCDIBehavior>
       <NavIndex>2</NavIndex>
       <ComIndex>2</ComIndex>
</Instrument>
```

3) In your interior model.xml file:
   a. Define two new input event sources for the indexed instruments:

```
<ModelBehaviors>
  <Include ModelBehaviorFile="ASOBO\Inputs\Helpers.xml"/>
  <InputEvent ID="AS430_1">
    <Presets>
      <Extend Target="ASOBO_GIE_Anim_Handling">
        <Parameters Type="Default">
          <INPUT_EVENT_ID_SOURCE>AS430_1</INPUT_EVENT_ID_SOURCE>
        </Parameters>
      </Extend>
    </Presets>
  </InputEvent>
  <InputEvent ID="AS430_2">
    <Presets>
      <Extend Target="ASOBO_GIE_Anim_Handling">
        <Parameters Type="Default">
          <INPUT_EVENT_ID_SOURCE>AS430_2</INPUT_EVENT_ID_SOURCE>
        </Parameters>
      </Extend>
    </Presets>
  </InputEvent>
</ModelBehaviors>
```

b. For each instrument, set the AS430 parameter to the indexed variant:

```
<Component ID="AS430_1">
  <Parameters Type="Default">
      <AS430>AS430_1</AS430>
    ...
  </Parameters>
  ...
</Component>

<Component ID="AS430_2">
  <Parameters Type="Default">
      <AS430>AS430_2</AS430>
    ...
  </Parameters>
  ...
</Component>
```

Configuring the GNS for support of hardware keyboard input:

The update GNS430/530 supports using a user's keyboard/mouse to input waypoint idents in the waypoint or direct to screens. This is activated by a user clicking with their mouse on a keyboard icon on the entry line, but requires that the screen allow for and capture mouse clicks.

Notes regarding multiple instruments per panel and "hot-swapping".

Many planes attempt to provide the user with a collection of user-selectable avionics configurations to fit individual preferences, such as supporting a GTN750 as a swappable option with a GNS series navigator ~~and needs~~. There are generally two ways that this is accomplished. One is to provide a different panel~~.xml~~ for each particular configuration. The second is to load all the instruments into one panel and use a selector of some sort to show or hide individual instrument nodes based upon the user-selected configuration.

With the WT GNS units the former will continue to work fine. However, developers who choose to implement the latter solution may find that that does not work with our new code. The reasons for this are complex, but essentially resolve to the fact that the Working Title avionics need to take complete control of the plane and its events and key mappings in order to function. This is the only way that we can accomplish what we do with these systems.

What this means is that our units may not function properly in a system which loads them alongside other avionics systems. The units could be in conflict over who has control of the flight plan, and who receives button presses and other user-interaction events. The interactions between avionics and the sim are complex and with the current functionality provided by the simulator we cannot support the "hot-swapping" model of instrument loading. Developers who have previously gone this route may wish to provide an alternate option that loads just the GNS units and leave hot-swapping for other instruments that do not have as complex an interaction with the sim.