Dashboard Live

Dokumentation / Manual

You find the english version *here*

1 Dashboards im Vanilla-Spiel

Es wird empfohlen, bei der Lektüre dieser Dokumentation die xml-Datei eines beliebigen Fahrzeugs zu öffnen, um darin die hier enthaltenen Beschreibungen nachvollziehen zu können.

1.1 Grundlagen

Im Vanilla-Spiel werden Informationen aus den Fahrzeugen und Geräten in der Kabine über Dashboards dargestellt. Die abzubildende Information ergibt sich dabei entweder aus einem festgelegten „`valueType`“, aus zugeordneten Gruppen, oder aus einer Kombination aus beidem.

ValueTypes werden jeweils in den Lua-Skripten der Spezialisierungen der Fahrzeuge definiert und in der xml-Datei des Fahrzeugs eingetragen.

Beispiele:

- `valueType="speed"`: Liefert die aktuelle Geschwindigkeit als Zahl (aus: „motorized“),
- `valueType="rpm"`: Liefert die aktuelle Motordrehzahl (aus „motorized“),
- `valueType="cruiseControl"`: Liefert die Geschwindigkeitseinstellung des Tempomaten (aus: „drivable“),
- `valueType="cruiseControlActive"`: Liefert den Status des Tempomaten (aus „drivable“).

Das Spiel unterscheidet mehrere Dashboard-Typen, die sogenannten „displayTypes“:

- `displayType="VISIBILITY"`: Dieses Element wird je nach Zustand ein- oder ausgeblendet.
- `displayType="EMITTER"`: Dieses Element wird je nach Zustand beleuchtet. Dabei kann sowohl für den inaktiven als auch für den aktiven Zustand eine Farbe festgelegt werden.
- `displayType="NUMBER"`: Dieses Element zeigt den jeweiligen Zustand als Zahlenwert an. Für die Zahl muss im Fahrzeug eine entsprechende TransformGroup existieren, die je abzubildender Ziffer einen Node enthalten muss.
- `displayType="TEXT"`: Dieses Element zeigt den jeweiligen Zustand als Text oder Zahl an. Für Zahlenwerte muss eine entsprechende Formatierung als Option angegeben werden.
- `displayType="ANIMATION"`: Dieses Element steuert eine Animation. Je nach Zustand wird die Animation zwischen 0% und 100% ausgeführt.
- `displayType="ROT"`: Dieses Element dreht den zugewiesenen Node je nach Zustand.
- `displayType="MULTI_STATE"`: Dieses Element kann mehrere Zustände haben, denen jeweils unterschiedliche Anzeigestatus zugewiesen werden können.

1.2 Gruppeneinträge

Gruppen werden zentral in der xml-Datei des Fahrzeugs in der Sektion `<dashboard>` festgelegt. Die hier definierten Gruppen können in allen Dashboard-Funktionseinträgen verwendet werden und haben dadurch gewissermaßen eine Filterfunktion.

Gruppendefinitionen befinden sich immer im Abschnitt `<dashboard>` der xml-Datei eines Fahrzeugs und haben den folgenden Aufbau:

```
<dashboard>
  <groups>
    <group name="NAME" filterFunktion="wert"/>
  </group>
</dashboard>
```

Der Name kann frei gewählt werden und sollte aus Großbuchstaben bestehen. Die Filterfunktion legt fest, wann die Gruppe aktiv ist und wann nicht. Definiert werden die möglichen Filterfunktionen in den Lua-Skripten der jeweiligen Spezialisierungen.

Beispiel:

```
<dashboard>
  <groups>
    <group name="MOTOR_STARTING" isMotorStarting="true"/>
  </groups>
</dashboard>
```

Diese Gruppe ist aktiv, wenn der Motor des Fahrzeugs startet oder läuft. Definiert wird „isMotorStarting“ in der Spezialisierung „motorized“.

1.3 Funktionseinträge

Funktionseinträge können sich an mehreren Stellen in der xml-Datei des Fahrzeugs befinden. Dabei wird unterschieden zwischen Standardeinträgen (ohne „`valueType`“-Eintrag) und spezialisierten Einträgen (mit „`valueType`“-Eintrag).

Die Standardeinträge dürfen nur in der Sektion `<dashboard><default>` vorkommen:

```
<dashboard>
  <default>
    <dashboard node="nodeName" displayType="EMITTER" […] groups="MOTOR_ACTIVE"/>
  </default>
</dashboard>
```

In diesem Beispiel leuchtet der Node „nodeName“ immer, wenn der Motor des Fahrzeugs läuft.

Spezialisierte Einträge stehen bei den Spezialisierungen, durch die sie definiert und umgesetzt werden. Beispiel:

```
<motorized>
  […]
  <dashboards>
    <dashboard valueType="speed" node="sNode" displayType="TEXT" […] groups="MOTOR_ACTIVE"/>
  </dashboards>
</motorized>
```

In diesem Beispiel wird am Node „sNode“ die aktuelle Geschwindigkeit ausgegeben, aber nur, wenn der Motor läuft. Dieser Eintrag darf nur im Abschnitt `<motorized>` stehen, weil der valueType „speed“ nur dort bekannt ist.

2 DashboardLive

2.1 Funktionsweise

DashboardLive nutzt die oben beschriebenen Mechanismen des Vanilla-Spiels und erweitert diese um zahlreiche Filterfunktionen für Gruppen und valueTypes für Funktionseinträge.

Außerdem wird eine neue Funktion eingeführt:

- `displayType="AUDIO"`: Anders als es der Name „`displayType`“ suggeriert, wird hier keine Anzeige, sondern eine Audio-Ausgabe angesteuert.
  Hierzu werden folgende Parameter erwartet:
  - `#audioFile`: Pfad zu der abzuspielenden Sound-Datei (ogg-Format)
  - `#audioName`: Eindeutiger Name (frei wählbar)
  - `#loop`: Wie oft soll die Sound-Datei abgespielt werden (1..n)
  - `#volume`: Die Lautstärke als Wert zwischen 0 und 1

  Beispiel:

```
<dashboard […] displayType="AUDIO" audioName="FLW_TRAILER1P1" loop="1"
audioFile="sounds/claasBeep.ogg" volume=”1” />
```

2.1.1 Dark Mode

Falls Anzeigeelemente oder Terminals für einen Dark Mode vorbereitet werden sollen, können zusätzlich zu den Farbwerten aus dem Spiel auch Farbwerte für einen Dark Mode festgelegt werden:

- `baseColorDarkMode`: Farbe eines Emitters, wenn dieser nicht aktiv ist.
- `emitColorDarkMode`: Farbe eines Emitters, wenn dieser aktiv ist.
- `intensityDarkMode`: Leuchtintensität.

Diese Farben können überall gesetzt werden, wo auch `baseColor`, `emitColor` und `intensity` festgelegt werden können.

2.2 Gruppeneinträge

Genau wie im Vanilla-Spiel werden auch die Gruppeneinträge für DashboardLive in der Gruppensektion der xml-Datei des Fahrzeugs vorgenommen:

```
<dashboard>

  <groups>

    <group name="NAME" filterFunktion="wert" op=”Modus” dbl=”Gruppenbefehl“
           dblAttacherJoindIndices="# # #" />

  </group>

</dashboard>
```

Modus kann den Wert “and” oder den Wert „or“ haben:

- „`and`“ bedeutet, dass der Gruppenbefehl logisch „und“-verknüpft wird, die Gruppe also nur aktiv ist, wenn die Filterfunktion und die über den Gruppenbefehl aufgerufene DashboardLive-Funktion beide aktiv sind.
- „`or`“ bedeutet, dass der Gruppenbefehl logisch „oder“ verknüpft wird, die Gruppe also bereits dann aktiv ist, wenn nur die Filterfunktion, nur die DashboardLive-Funktion oder beide aktiv sind.

Gruppenbefehl benennt die DashboardLive-Funktion, die genutzt werden soll

dblAttacherJointIndices definiert, auf welche attacherJoints des Fahrzeugs die Funktion angewendet werden soll.

Beispiel:

```
<dashboard>

  <groups>

    <group name="DBL_LOWERALE" filterFunktion="isMotorRunning” op=”and” dbl=”base_lowerable”
           dblAttacherJoindIndices="1 2 3" />

  </group>

</dashboard>
```

Diese Gruppe ist aktiv, wenn der Motor des Fahrzeugs läuft und das an AttacherJoint 1, 2, oder 3 angehängte Gerät absenkbar ist.

Tipp: Meist ist die Nutzung der DashboardLive-Funktionen über Funktionseinträge einfacher, daher sollte mit DashboardLive-Gruppen eher sparsam umgegangen werden.

2.3 Funktionseinträge

Die Funktionseinträge von DashboardLive werden alle in der Sektion `<dashboard><dashboardLive>` gesetzt und funktionieren auf die gleiche Art und Weise wie die Funktionseinträge im Vanilla-Spiel.

```
<dashboard>

  <dashboardLive>

    <dashboard valueType=”vT” cmd=”c” […] node=“node“ displayType="EMITTER" […] />

  </dashboardLive>

</dashboard>
```

Hierbei legt „vT“ das Modul fest, das abgefragt wird und „c“ gibt den Befehl mit, der für die Abfrage genutzt wird.

Das liest sich hier komplizierter, als es ist, deshalb ein Beispiel:

```
<dashboard>

  <dashboardLive>

    <dashboard valueType="base" cmd="lifted" joints="6" displayType="VISIBILITY" node="icon"
               groups="MOTOR_ACTIVE" />

  </dashboardLive>

</dashboard>
```

Dieser Befehl ruft das Modul „`base`“ auf, das Funktionen des Vanilla-Spiels bereitstellt, und nutzt die Funktion „`lifted`“. Diese Funktion ermittelt, ob das Gerät an attacherJoint 6 des Fahrzeugs ausgehoben ist. In diesem Fall wird node „icon“ sichtbar, aber nur, wenn der Motor läuft.

Hinweis: Immer wenn in „joints“ mehrere attacherJoints aufgelistet werden, kann zusätzlich über „option“ festgelegt werden, wie die Auswertung erfolgen soll:

- `option=“all“`: Die Abfrage ist nur aktiv, wenn sie für alle attacherJoints zutrifft
- `option=“any“`: Die Abfrage ist aktiv, sobald sie für mindestens ein attacherJoint zutrifft.

Beispiel:

```
<dashboard>

  <dashboardLive>

    <dashboard valueType="base" cmd="lowered" joints="1 2 3" option=”any” node="icon"
              displayType="VISIBILITY" groups="DBL_LOWERABLE" />

  </dashboardLive>

</dashboard>
```

Diese Zeile zeigt Node „icon“ nur an, wenn mindestens ein attacherJoint abgesenkt ist, aber nur, wenn auch die Gruppe „DBL_LOWERABLE“ aktiv ist. Zusätzlich ist es seit der Version 1.3.0.0 von DashboardLive möglich, weitere Bedingungen festzulegen, die erfüllt sein müssen, damit ein Element aktiv wird. Dies erfolgt über `cond="Bedingung"` und `condValue="Vergleichswert"`:

cond=	Bedeutung
less	Aktiv, wenn der entsprechende Wert kleiner als condValue ist
lessequal	Aktiv, wenn der entsprechende Wert kleiner oder gleich condValue ist
equal	Aktiv, wenn der entsprechende Wert gleich condValue ist
moreequal	Aktiv, wenn der entsprechende Wert größer oder gleich condValue ist
more	Aktiv, wenn der entsprechende Wert größer als condValue ist
not	Dreht das Ergebnis um: Aktiv wird inaktiv / inaktiv wird aktiv)

Beispiel:

```
<dashboard>

  <dashboardLive>

    <dashboard valueType="base" cmd="liftState" joints="6" cond=”more” condValue=”0.9”
displayType="VISIBILITY" node="icon" groups="MOTOR_ACTIVE" />

  </dashboardLive>

</dashboard>
```

Diese Zeile zeigt Node „icon“ nur an, wenn das Gerät an attacherJoint 6 mehr als 90% ausgehoben ist.

Eine vollständige Übersicht aller DashboardLive-Befehle ist hier zu finden: DBL-Commands. Hieraus kann auch entnommen werden, für welche Funktionen `cond` und `condValue` genutzt werden können.

Einige Befehle erfordern noch weitere Angaben über weitere Einträge („state“, „trailer“, etc.). Diese Angaben können ebenfalls der Übersicht entnommen werden.

1 Dashboards in the Vanilla Game

While reading this documentation, it is recommended to open the xml file of any vehicle in order to be able to follow the descriptions contained herein.

1.1 Basics

In the vanilla game, information from the vehicles and devices in the cabin is displayed via dashboards. The information to be displayed results either from a defined "`valueType`", from assigned groups, or from a combination of both.

ValueTypes are each defined in the Lua scripts of the vehicles' specialisations and entered in the vehicle's xml file.

Examples:

- `valueType="speed"`: Delivers the current speed as a number (from: "motorized"),
- `valueType="rpm"`: Delivers the current engine speed (from "motorized"),
- `valueType="cruiseControl"`: Delivers the speed setting of the cruise control (from: "drivable"),
- `valueType="cruiseControlActive"`: Delivers the status of the cruise control (from: "drivable").

The game distinguishes several dashboard types, the so-called "displayTypes":

- `displayType="VISIBILITY"`: This element is shown or hidden depending on the state.
- `displayType="EMITTER"`: This element is illuminated depending on the state. A colour can be defined for both the inactive and the active state.
- `displayType="NUMBER"`: This element displays the respective state as a numerical value. For the number, a corresponding TransformGroup must exist in the vehicle, which must contain a node for each digit to be displayed.
- `displayType="TEXT"`: This element displays the respective state as text or number. For numerical values, a corresponding formatting must be specified as an option.
- `displayType="ANIMATION"`: This element controls an animation. Depending on the state, the animation is executed between 0% and 100%.
- `displayType="ROT"`: This element rotates the assigned node depending on the state.
- `displayType="MULTI_STATE"`: This element can have multiple states, each of which can be assigned different display states.

1.2 Group entries

Groups are defined centrally in the vehicle's xml file in the <dashboard> section. The groups defined here can be used in all dashboard function entries and thus have a filter function, so to speak.

Group definitions are always located in the <dashboard> section of a vehicle's xml file and have the following structure:

```
<dashboard>

  <groups>

    <group name="NAME" filterFunction="value"/>

  </group>

</dashboard>
```

The name can be freely chosen and should consist of capital letters. The filterFunction determines when the group is active and when it is not. The possible filter functions are defined in the Lua scripts of the respective specializations.

Example:

```
<dashboard>
  <groups>
    <group name="MOTOR_STARTING" isMotorStarting="true"/>
  </groups>
</dashboard>
```

This group is active when the engine of the vehicle starts or runs. Defined is "isMotorStarting" in the specialization "motorized".

1.3 Function entries

Function entries can be located in several places in the xml file of the vehicle. A distinction is made between standard entries (without "`valueType`" entry) and specialized entries (with "`valueType`" entry).

The standard entries may only occur in the `<dashboard><default>` section:

```
<dashboard>
  <default>
    <dashboard node="nodeName" displayType="EMITTER" [...] groups="MOTOR_ACTIVE"/>
  </default>
</dashboard>
```

In this example, the node "nodeName" is always lit when the vehicle's engine is running.

Specialized entries are at the specializations by which they are defined and implemented. Example:

```
<motorized>
  [...]
  <dashboards>
    <dashboard valueType="speed" node="sNode" displayType="TEXT" [...]
               groups="MOTOR_ACTIVE"/>
  </dashboards>
</motorized>
```

In this example, the current speed is output at the node "sNode", but only when the motor is running. This entry may only be in the `<motorized>` section because the valueType "speed" is only known there.

2 DashboardLive

2.1 Functionality

DashboardLive uses the mechanisms of the vanilla game described above and extends them with numerous filter functions for groups and valueTypes for function entries.

A new function is also introduced:

- `displayType="AUDIO"`: Contrary to what the name "displayType" suggests, an audio output is triggered here instead of a display.
  The following parameters are expected for this:
  - `#audioFile`: Path to the sound file to be played (ogg format)
  - `#audioName`: Unique name (freely selectable)
  - `#loop`: How often the sound file should be played (1..n)
  - `#volume`: The volume as a value between 0 and 1

  Example:

```
<dashboard [...] displayType="AUDIO" audioName="FLW_TRAILER1P1" loop="1"
audioFile="sounds/claasBeep.ogg" volume="1" />
```

2.1.1 Dark Mode

If display elements or terminals are to be prepared for dark mode, colour values for dark mode can be defined in addition to the colour values from the game:

- `baseColorDarkMode`: Colour of an emitter if it is not active.
- `emitColorDarkMode`: Colour of an emitter when it is active.
- `intensityDarkMode`: Light intensity.

These colours can be set anywhere where baseColor, emitColor and intensity can also be set.

2.2 Group entries

Just like in the vanilla game, the group entries for DashboardLive are made in the group section of the vehicle xml file:

```
<dashboard>
  <groups>
    <group name="NAME" filterFunction="value" op="mode" dbl="group command"
           dblAttacherJoindIndices="# #" />
  </group>
</dashboard>
```

Mode can have the value "and" or the value "or":

- "`and`" means that the group command is logically "and"-linked, i.e. the group is only active if the filter function and the DashboardLive function called via the group command are both active.

- "`or`" means that the group command is logically linked with "or", i.e. the group is already active when only the filter function, only the DashboardLive function or both are active.

“group command” names the DashboardLive function to be used.

“dblAttacherJointIndices” defines to which attacherJoints of the vehicle the function should be applied.

Example:

```
<dashboard>

  <groups>

    <group name="DBL_LOWERALE" filterFunction="isMotorRunning" op="and"
           dbl="base_lowerable" dblAttacherJoindIndices="1 2 3" />

  </groups>

</dashboard>
```

This group is active when the vehicle's engine is running and the unit attached to AttacherJoint 1, 2, or 3 is lowerable.

Tip: It is usually easier to use DashboardLive functions via function entries, so DashboardLive groups should be used sparingly.

2.3 Function entries

DashboardLive function entries are all set in the <dashboard><dashboardLive> section and work in the same way as function entries in the vanilla game.

```
<dashboard>

  <dashboardLive>

    <dashboard valueType="vT" cmd="c" [...] node="node" displayType="EMITTER" [...] />

  </dashboardLive>

</dashboard>
```

Here, "vT" specifies the module that is queried and "c" specifies the command that is used for the query.

This reads more complicated than it is, so here is an example:

```
<dashboard>

  <dashboardLive>

    <dashboard valueType="base" cmd="lifted" joints="6" displayType="VISIBILITY" node="icon"

               groups="MOTOR_ACTIVE" />

  </dashboardLive>

</dashboard>
```

This command calls the module "`base`", which provides functions of the vanilla game, and uses the function "`lifted`". This function determines if the device is lifted at attacherJoint 6 of the vehicle. In this case, node "icon" becomes visible, but only when the engine is running.

Note: Whenever several attacherJoints are listed in "joints", "option" can also be used to specify how the evaluation is to be carried out:

- `option="all"`: The query is only active if it is true for all attacherJoints.

- `option="any"`: The query is active as soon as it applies to at least one attacherJoint.

Example:

```
<dashboard>

  <dashboardLive>

    <dashboard valueType="base" cmd="lowered" joints="1 2 3" option=”any” node="icon"
               displayType="VISIBILITY" groups="DBL_LOWERABLE" />

  </dashboardLive>

</dashboard>
```

This line displays node "icon" only if at least one attacherJoint is lowered, but only if the group "DBL_LOWERABLE" is also active.

Since version 1.3.0.0 of DashboardLive, it is also possible to define additional conditions that must be met for an element to become active. This is done via `cond="condition"` and `condValue="comparison value"`:

cond=	meaning
less	Active if the corresponding value is less than condValue
lessequal	Active if the corresponding value is less than or equal to condValue
equal	Active if the corresponding value is equal to condValue
moreequal	Active if the corresponding value is greater than or equal to condValue
more	Active if the corresponding value is greater than condValue
not	Reverses the result: Active becomes inactive / inactive becomes active)

Example:

```
<dashboard>

  <dashboardLive>

    <dashboard valueType="base" cmd="liftState" joints="6" cond="more"
      condValue="0.9" displayType="VISIBILITY" node="icon" groups="MOTOR_ACTIVE" />

  </dashboardLive>

</dashboard>
```

This line only displays node "icon" if the device at attacherJoint 6 is more than 90% raised.

A complete overview of all DashboardLive commands can be found here: DBL-Commands.

Some commands require additional information by further entries ("state", "trailer", etc.). These needed entries can also be taken from the overview.