Message Bus (bpy.msgbus)

The message bus system can be used to receive notifications when properties of Blender datablocks are changed via the data API.

Limitations

The message bus system is triggered by updates via the RNA system. This means that the following updates will result in a notification on the message bu

- Changes via the Python API, for example `some_object.location.x += 3`.
- Changes via the sliders, fields, and buttons in the user interface.

The following updates do **not** trigger message bus notifications:

- Moving objects in the 3D Viewport.
- Changes performed by the animation system.

Example Use

Below is an example of subscription to changes in the active object’s location.

```
import bpy

# Any Python object can act as the subscription's owner.
owner = object()

subscribe_to = bpy.context.object.location


def msgbus_callback(*args):
    # This will print:
    # Something changed! (1, 2, 3)
    print("Something changed!", args)


bpy.msgbus.subscribe_rna(
    key=subscribe_to,
    owner=owner,
    args=(1, 2, 3),
    notify=msgbus_callback,
)
```

Some properties are converted to Python objects when you retrieve them. This needs to be avoided in order to create the subscription, by using `datablock.path_resolve("property_name", False)`:

```
subscribe_to = bpy.context.object.path_resolve("name", False)
```

It is also possible to create subscriptions on a property of all instances of a certain type:

```
subscribe_to = (bpy.types.Object, "location")
```

bpy.msgbus.**clear_by_owner(owner)**

Clear all subscribers using this owner.

bpy.msgbus.**publish_rna(key)**

PARAMETERS:

key (`bpy.types.Property` | `bpy.types.Struct` | tuple[`bpy.types.Struct`, str]) –

Represents the type of data being subscribed to

Arguments include - A property instance. - A struct type. - A tuple representing a (struct, property name) pair.

Notify subscribers of changes to this property (this typically doesn't need to be called explicitly since changes will automatically publish updates). Ir some cases it may be useful to publish changes explicitly using more general keys.

bpy.msgbus.**subscribe_rna(key, owner, args, notify, options=set())**

Register a message bus subscription. It will be cleared when another blend file is loaded, or can be cleared explicitly via `bpy.msgbus.clear_by_owner()`.

PARAMETERS:

- **key** (`bpy.types.Property` | `bpy.types.Struct` | tuple[`bpy.types.Struct`, str]) –

  Represents the type of data being subscribed to

  Arguments include - A property instance. - A struct type. - A tuple representing a (struct, property name) pair.

- **owner** (*Any*) – Handle for this subscription (compared by identity).
- **options** (*set[str]*) –

  Change the behavior of the subscriber.

  - `PERSISTENT` when set, the subscriber will be kept when remapping ID data.

Note

All subscribers will be cleared on file-load. Subscribers can be re-registered on load, see `bpy.app.handlers.load_post`.