Cloud \ Dynamic Attributes

The **Dynamic Cloud Attribute** nodes save and read small pieces of text in SimLab's cloud, so a VR experience can remember information between sessions and across devices — a person's progress, a saved choice, a score, and so on.

They are the flexible counterpart to the older **Get Cloud Attribute** and **Set Cloud Attribute** nodes. The difference is how a saved value is identified. The older nodes use a single fixed **name** that you type into the node. These nodes instead identify each value by a **Key** — a structured value (a JSON object) that can hold one field or several, which you build with the JSON Object nodes. Because a key can carry several pieces of information at once (for example a category *and* a level), one node can manage a whole family of related values, and the **Search** node can find many of them at once by matching only part of a key.

What's on this page

- **Set Dynamic Cloud Attribute** — save a value, creating it or replacing what was there.
- **Soft Set Dynamic Cloud Attribute (String)** — save a value only if nothing is stored under that key yet.
- **Get Dynamic Cloud Attribute** — read back the value saved under a key.
- **Search Dynamic Cloud Attribute** — find every value whose key matches the fields you give.
- **Check Dynamic Cloud Attribute Exists** — check whether a value is saved under a key, without reading it.
- **Remove Dynamic Cloud Attribute** — delete the value saved under a key.

How these nodes work

All six nodes share the same few ideas.

The Key

Every value is filed under a **Key** — a JSON object you build with the JSON Object nodes. A key can hold a single field or several (for example `{ "category": "progress", "level": "2" }`). To read, check, or remove a value later, give the same Key you saved it under. A Key cannot be empty.

Who the value belongs to

Whose value a node reads or saves is decided by the scene's **Dashboard Project** (set in SimLab Composer from the VR menu), not by the nodes themselves. The project's **Data Owner** setting applies to all six nodes:

- **Me** — everyone who runs the experience shares one value for each Key, so a value saved by one person is the value everyone else reads.
- **User** — each person running the experience has their own separate value for each Key.

SimLab Composer won't export a scene that uses these nodes until the scene has a Dashboard Project.

They talk to the cloud

Because these nodes contact a server, they wait for the cloud to answer before carrying on: the **Execute** output fires only once the cloud has replied — whether the request worked or not. Every node has a **Success** output that is true only when the request genuinely worked; always send it into a **Branch** and trust the result only on the "true" side. The experience must be running from a signed-in account for cloud data to work.

Saving values

Set Dynamic Cloud Attribute

Saves a piece of text in SimLab’s cloud so your experience can remember it later, even after the person closes it or switches to another device.

What it does

This node stores the text you give it under a **Key** that you build with the JSON Object nodes. The Key is how the saved value is found again later — instead of one fixed name, you can put several pieces of information into the Key (such as a category and a level) so that one node can manage a whole family of saved values.

If nothing has been saved under that Key yet, a new value is created. If something is already saved under that exact same Key, its text is replaced with the new text. Other saved values, under different Keys, are left untouched. Because this node talks to the cloud, it waits for the cloud to answer before continuing, and it reports whether the save genuinely worked through the **Success** output. For cloud data to work, the experience must be running from a signed-in account.

Inputs

Port	Type	What to connect
Execute	Trigger	Wire this from the previous node’s Execute output.
Key	JSON object	The Key that identifies this saved value, built with the JSON Object nodes. It can hold one or several name/value fields. The Key cannot be empty.
Value	Text	The text you want to save under this Key.

Outputs

Port	Type	What you get
Execute	Trigger	Fires once the cloud has answered, whether the save worked or not. Always check **Success** before trusting the result.

Port	Type	What you get
Success	True / false	True only when the value was genuinely saved; false if something went wrong. Wire this into a Branch so your experience can react to either case.

Example

Key input	a JSON object like `{ "category": "progress", "level": "2" }`
Value input	`completed`
Success output	`true` once the value has been saved

Tips

- Build the Key with the JSON Object nodes before this node runs, and use the same Key later when you want to read the value back.
- To play it safe, send the **Success** output into a Branch — continue only when it is true, and show a message or retry when it is false.
- Saving again with the very same Key replaces the old text, so you can use this node to keep an up-to-date value (like a person’s latest progress).

Soft Set Dynamic Cloud Attribute (String)

Saves a piece of text in the cloud under a key you build yourself — but only the first time, so it never overwrites a value that is already stored there.

What it does

This node stores a bit of text in SimLab’s cloud so your experience can remember it between sessions and across devices. Instead of a fixed name typed into the node, the saved value is identified by a **Key** — a JSON object you build with the JSON Object nodes. The key can carry one field or several (for example a category and a level), so a single node can manage a whole family of saved values.

The word “Soft” is the important part: this node only writes when nothing is saved under that key yet. If a value is already there, it is left exactly as it is — nothing is overwritten. Use this when you want to set a starting value once and then leave it alone. Only the value under this one key is affected; everything else you have saved stays untouched.

Because the node talks to the cloud, it waits for the cloud to answer before moving on. Always check the **Success** result (wire it into a Branch) before trusting that the value is in place. A `true` result means that after this node the key holds a value — either the text you just wrote, or the one that was already there. The experience must be running from a signed-in account for cloud data to work.

Inputs

Port	Type	What to connect
Execute	Trigger	Wire this from the previous node’s Execute output to run the node.
Key	JSON object	The key that identifies this saved value — a JSON object you build with the JSON Object nodes. It can hold one field or several (for example a category and a level). It cannot be empty.
Value	Text	The text to save — but only if nothing is stored under this key yet.

Outputs

Port	Type	What you get
Execute	Trigger	Fires once the cloud has answered, whether the call worked or not. Check Success before continuing.
Success	True / false	True when the key holds a value after this node runs — either the text you just saved, or the one that was already there. False means the cloud could not be reached or you are not signed in.

Example

Key input	a JSON object such as `{ "category": "progress", "level": "1" }`
Value input	`not started`
Success output	`true` — the starting status is now saved (or was already saved from a previous run)

Tips

- Use this node to set a one-time starting value, like a person's initial progress. If you instead want to change a value every time, use the **Set Dynamic Cloud Attribute** node, which always overwrites.
- Build the Key with the JSON Object nodes before this node runs, and keep it consistent so you can read the same value back later.
- Always wire Success into a Branch so your experience can react if the cloud could not be reached.

Reading and finding values

Get Dynamic Cloud Attribute

Reads back a piece of saved text from SimLab's cloud, looking it up by the key it was saved under.

What it does

This node fetches a value that was previously saved to the cloud and hands it back to you as text. Instead of identifying the saved value by a fixed name typed into the node, you identify it with a **Key** — a JSON object you build with the JSON Object nodes. Because a key can hold one or several name/value fields, a single node can look up a whole family of saved values (for example, the saved status for a different level each time).

Reading a value never changes it — this node only looks, it doesn't save anything, and other saved values are left untouched. Because it talks to a server, the node waits for the cloud to answer before it continues. The **Execute** output fires once the cloud has replied, whether or not it found anything, so always check the **Success** output (wire it into a Branch) before trusting the result. The experience must be running from a signed-in account for cloud data to work.

Inputs

Port	Type	What to connect
Execute	Trigger	Wire this from the previous node's Execute output.
Key	JSON object	The key that identifies the saved value you want to read. Build it with the JSON Object nodes — it can carry one or several name/value fields. It must match the key the value was saved under, and it cannot be empty.

Outputs

Port	Type	What you get
Execute	Trigger	Fires once the cloud has answered, whether the call worked or not. Continue your flow from here.
Success	True / false	`true` only when the value was found and read back successfully. It is `false` if the call failed or if no saved value matched the key. Check this before using the result.
Value	Text	The saved text that was read back. It is empty when Success is `false`.

Example

Key input	`{ "category": "progress", "level": "3" }`
Success output	`true`
Value output	`completed`

Tips

- Build the same key here that you used when saving the value with the **Set Dynamic Cloud Attribute** node — if the key doesn't match, nothing is found and Success comes back `false`.

- Always send the Success output into a Branch so your flow can react cleanly when a value is missing or the cloud couldn’t be reached.

🔍 Search Dynamic Cloud Attribute

Finds every value saved in SimLab’s cloud whose key matches the fields you give it, so you can look up a whole group of related saved values at once.

What it does

Each value you save in the cloud is filed under a Key — a set of name/value fields you build with the JSON Object nodes (for example a category and a level). This node lets you search by only *part* of a key: you supply a Partial Key with just the fields you care about, and it returns every saved value whose key contains those fields. For example, give it just the category and you get back every value saved under that category, across all the levels.

You get back two matching lists: one with the full keys of everything it found, and one with the saved text for each, lined up in the same order — the first key goes with the first value, the second with the second, and so on. Because this node talks to the cloud, it waits for the cloud to answer before moving on, and it tells you through the Success output whether the lookup actually worked. Searching only reads your saved values; it never changes them.

Inputs

Port	Type	What to connect
Execute	Trigger	Wire this from the previous node’s Execute output.
Partial Key	JSON object	The fields to search for, built with the JSON Object nodes. Include only the parts of the key you want to match on — every saved value whose key contains these fields is returned. This must not be empty.

Outputs

Port	Type	What you get

Execute	Trigger	Fires once the cloud has answered, whether the search worked or not. Always check Success before using the result.
Success	True / false	True only when the cloud genuinely answered the search. If it is false, both lists come back empty.
Keys	JSON array	A list of the full keys of every saved value that matched, one entry per match.
Values	JSON array	A list of the saved text for each match, in the same order as Keys — the first value goes with the first key, and so on.

Example

Partial Key input	a JSON object holding just `{ "category": "progress" }` — matching every value saved under that category
Success output	`true`
Keys output	a list such as `[ {"category":"progress","level":"1"}, {"category":"progress","level":"2"} ]`
Values output	a list such as `[ "completed", "in progress" ]` — lined up with the keys above

Tips

- Wire Success into a Branch node and only read the Keys and Values lists when it is true.
- The two lists always match up by position, so step through them together when you process the results.
- The more fields you put in the Partial Key, the narrower the search; include fewer fields to find a wider group of saved values.

Check Dynamic Cloud Attribute Exists

Asks SimLab's cloud whether a saved value with a given key already exists, without reading or changing it.

What it does

This node looks up whether a value you previously saved in the cloud is there or not. You hand it a Key — a JSON object you build with the JSON Object nodes — and it reports back "yes, something is saved under this key" or "no, nothing is saved here yet." It does not return the saved value itself, and it never changes anything; it only checks.

Because this node talks to the cloud, it waits for the cloud to answer before moving on. The result comes back in two parts: **Success** tells you whether the check itself worked, and **Exists** tells you whether a value was found. Always check **Success** first (wire it into a Branch) — if the call did not work, the **Exists** answer cannot be trusted. The experience must be running from a signed-in account for cloud data to work.

Inputs

Port	Type	What to connect
Execute	Trigger	Wire this from the previous node's Execute output to run the check.
Key	JSON object	The key that identifies the saved value you want to look for. Build it with the JSON Object nodes — it can hold one or several name/value fields. It must not be empty.

Outputs

Port	Type	What you get
Execute	Trigger	Fires once the cloud has answered, whether the check worked or not. Continue your flow from here.
Success	True / false	True only when the check genuinely worked. If it is false (for example, the account was not signed in), ignore the Exists result.

Port	Type	What you get
Exists	True / false	True if a saved value was found for this key, false if nothing is saved under it yet. Only meaningful when Success is true.

Example

Key input	a JSON object like `{ "category": "progress", "level": "fire-safety-1" }`
Success output	`true` — the cloud answered correctly
Exists output	`true` — progress is already saved for this level

Tips

- Use this node to decide whether to load existing progress or start fresh — check Exists, then branch to either read the saved value or save a new one.
- Build the same Key the same way every time you save and check, so the check finds the right value.

Removing values

Remove Dynamic Cloud Attribute

Deletes one saved value from the cloud — the one whose Key you provide.

What it does

Every Dynamic Cloud value is saved under a Key — a bundle of name/value fields you build with the JSON Object nodes (for example a category and a level). This node looks up the value saved under the Key you give it and removes it from the cloud. Only that one value is affected; every other saved value stays exactly as it was.

Because this node talks to the cloud, it waits for the cloud to answer before it continues. When the answer comes back, the **Success** output tells you whether it worked. Deleting a value that was already gone still counts as a success — the end result is the same: there is nothing saved under that Key. A real problem (such as not being signed in) gives

Success as false. Always send Success into a Branch and only trust the outcome when it is true. The experience must be running from a signed-in account for cloud data to work.

Inputs

Port	Type	What to connect
Execute	Trigger	Wire this from the previous node's Execute output.
Key	JSON object	The Key that identifies the saved value you want to remove. Build it with the JSON Object nodes — it can hold one field or several. It must match the Key the value was saved under, and it cannot be empty.

Outputs

Port	Type	What you get
Execute	Trigger	Fires once the cloud has answered — whether the removal worked or not. Continue your steps from here.
Success	True / false	True when the value was removed (or was already gone), false if something went wrong, such as not being signed in. Check this before trusting the result.

Example

Key input	`{ "category": "progress", "level": "3" }`
Success output	`true` — the saved progress for that level has been removed

Tips

- The Key here must match the one used when the value was saved. If the fields don't line up, the cloud finds nothing to remove.
- Removing a value that doesn't exist is safe — Success is still true, so you don't need to check first.

- Wire Success into a Branch so your experience can react if the removal couldn’t be completed.

Revision #2
Created 2026-06-14 12:13:05 UTC by Rafat
Updated 2026-10-25 08:21:54 UTC by Rafat