



Simutrans Squirrel 3.2

Script API Migration, Compatibility and Developer Reference

Product	Simutrans Standard	Squirrel runtime	3.2 (SQUIRREL_VERSION_NUMBER 320)
Integration revisions	r12165 – r12168 (trunk)	Completion revision	r12168
Documentation status	Current as of the integration revision	Document version	1.0
Date	2026-08-16	Document type	Developer Reference
Language	English — Canonical source		
Audience	Script authors · Scenario authors · AI authors · Simutrans developers		

Prepared from the validated Simutrans Standard Squirrel 3.2 migration.

EXECUTIVE SUMMARY

Simutrans Standard now embeds **Squirrel 3.2**. The embedded Squirrel sources were updated from a 2021 snapshot of the 3.1.x development line to the upstream [v3.2](#) release.

The important points for anybody who writes or maintains Simutrans scripts:

- This is an update of the **language runtime**, not a redesign of the Simutrans Script API.
- The migration was deliberately designed to **preserve the public Script API**. No registered function was added, removed or renamed by the runtime update, and no parameter list or default-argument behaviour changed.
- **Existing compatible scenarios and AI scripts do not need wholesale rewrites.**
- Several **internal VM integration points** changed, principally the C++ instance lookup call, closure creation for environment binding, and the string hash function.
- One **observable semantic difference** exists for scripts: the order in which the entries of a table are visited. It is documented in section 6 and it is the single thing worth reviewing in an existing script.
- **Old savegames containing script state were explicitly tested** and load correctly.
- Two new language/library conveniences arrived with the runtime: an inline form of environment binding, and `table.map`.

If you maintain a script and read only one section of this document, read [section 6](#).

Contents

1.	Executive summary	1
2.	Scope and terminology	3
3.	What changed	3
4.	What did not change	5
5.	Compatibility matrix	5
6.	Table iteration order	6
7.	Migration guide for script authors	7
8.	Scenario authors	8
9.	AI authors	8
10.	Paksets	9
11.	Examples	9
12.	Notes for Simutrans developers: the embedding layer	10
13.	Script API stability	12
14.	Validation summary	12
15.	Known limitations and notes	13
16.	Where to find what	14
17.	Availability and changelog	14
18.	Document history	15

2. Scope and terminology

This document uses three terms precisely, and the distinction matters throughout:

Term	Meaning
Squirrel	The embedded scripting <i>language and its runtime</i> , an external project vendored into Simutrans.
Script API	Simutrans's own registered interface — the classes, methods and constants that Simutrans exposes to scripts, documented in the Script API reference.
Simutrans Standard	The main Simutrans line, as distinct from Simutrans Extended.

"Squirrel 3.2" therefore describes the runtime. It does not describe a new version of the Script API, which is versioned separately and is unchanged by this work.

Runtime provenance

The Squirrel sources shipped with Simutrans are a vendored copy of the upstream project at [albertodemichelis/squirrel](https://github.com/albertodemichelis/squirrel).

	Commit	Description
Previous vendored basis	<code>23a0620658714b996d20da3d4dd1a0dcf9b0bd98</code>	A snapshot of the 3.1.x development line, dated 2021-09-16
New basis	<code>f92bc298784ceea459b12e2de33bdf672bfeb83</code>	Upstream release tag <code>v3.2</code> , dated 2022-02-10

The previous snapshot was taken from the development line *after* the 3.1 release, so it was neither an exact upstream 3.1 nor a 3.2. Describing the update simply as "3.1 to 3.2" would be inaccurate, which is why the commits above are quoted instead.

3. What changed

3.1 Summary

Change	Visible to scripts?	Visible to C++ developers?
Vendored Squirrel sources updated to upstream v3.2	indirectly	yes
Instance lookup API gained a parameter	no	yes
Inline <code>bindenv</code> syntax added to the language	yes, as a new option	yes

Change	Visible to scripts?	Visible to C++ developers?
<code>table.map</code> added to the standard library	yes, as a new option	no
String hash function changed	yes — see section 6	yes

3.2 Instance lookup API

Upstream Squirrel 3.2 changed the signature of the C API function that recovers a native instance pointer from the VM stack. It gained a fifth parameter that controls whether a type mismatch raises a Squirrel error or is reported to the caller as a plain failure:

```
/* before */
SQRESULT sq_getinstanceup(HSQIRRELVM v, SQInteger idx, SQUserPointer *p, SQUserPointer typetag);

/* Squirrel 3.2 */
SQRESULT sq_getinstanceup(HSQIRRELVM v, SQInteger idx, SQUserPointer *p, SQUserPointer typetag,
SQBool throwerror);
```

Every Simutrans call site was updated to pass the value that preserves its existing behaviour. **Script authors are not affected**: this is a C++ signature, not a Script API entry point, and the observable behaviour of the affected calls is unchanged.

Simutrans call sites touched: `src/simutrans/script/api/api_map_objects.cc`, `src/simutrans/script/api_class.h`, `src/squirrel/sq_extensions.cc`, plus the vendored standard-library files.

3.3 `bindenv` and the inline environment form

Binding an environment to a function is a feature of the Squirrel language, not a Simutrans class or Script API registration.

The existing `bindenv` method is **unchanged and continues to work**. Squirrel 3.2 adds an inline form in which the environment is written in square brackets between the function keyword and the parameter list. Both forms are shown in [section 11](#).

The inline form is a convenience. It does not give access to anything the method form did not already allow, and **existing scripts need no change**.

Internally this required the closure-creation opcode to carry the bound target, so the compiler and the VM were both updated.

3.4 `table.map`

Squirrel 3.2 adds a `map` method to tables in the standard library. Arrays already had one.

Two properties are worth knowing before using it:

- `table.map` returns an **array** of the callback results, not a table. This differs from `table.filter`, which returns a table.
- The results appear in table iteration order, so the order of the returned array is subject to [section 6](#). Sort it if the order matters.

3.5 String hash function

Upstream changed the hash function applied to string keys. This is the root cause of the one script-visible semantic difference in this migration, and it is covered in detail in [section 6](#).

4. What did not change

This section is deliberately explicit, because it is the part most script authors need.

- **The Simutrans Script API is unchanged.** No class name changed. No public method changed. No registration name changed. No parameter list or default-argument behaviour changed. This was measured, not assumed — see [section 13](#).
- **The scenario and AI entry points are unchanged.**
- **The savegame format is unchanged.**
- **The way scripts are loaded, suspended and resumed is unchanged.**
- **No pakset change of any kind is required** — see [section 10](#).

5. Compatibility matrix

Area	Compatibility status	Developer action
Existing Script API names	Unchanged. Registration count and calling conventions measured identical before and after.	None.
Scenario scripts	Compatible , subject to the iteration-order caveat.	Test on a current build; review any table iteration that makes a choice.
AI scripts	Compatible. The two shipped scripted AI players were run against the new runtime and remained compatible.	Same as scenarios. Expect scheduling and route choices to differ in detail — see section 9.
Saved games with script state	Old saves load correctly , tested with the shipped AI players. Save-and-reload under 3.2 also tested.	None. Back up before large-scale testing, as always.
Reverse direction: 3.2 save into an older build	Not claimed. Not part of this work and not tested.	Do not rely on it.
VM bindings (C++ embedding layer)	Changed. Instance lookup gained a parameter; closure creation carries a bound target.	Applies only to code that touches <code>src/squirrel/</code> or the Script API registration layer. See section 12.
Table iteration order	Changed. Different string hash function, therefore a different <code>foreach</code> order.	Review any script that relies on the order a table happens to produce. See section 6.

Area	Compatibility status	Developer action
Array iteration order	Unchanged. Arrays are ordered containers; their order is part of their contract.	None.
Error diagnostics	No new Squirrel errors observed in the validation campaign.	None.
Pakset assets (<code>.dat</code> , objects, graphics)	Unaffected.	None.

Note on scope. The compatibility statements above describe the scripts and code paths covered by the validation campaign — the scripted AI players shipped with Simutrans, and the savegame directions listed. They are not a guarantee that every possible third-party script is unaffected.

6. Table iteration order

CAUTION

Do not rely on the iteration order of Squirrel tables unless it is explicitly documented by the API.

Squirrel 3.2 changes the hash function used for string keys. Tables are hash containers, so this changes the order in which `foreach` visits their entries.

6.1 What this actually means

The consequence for a script is indirect but real. If a script iterates a table to pick the next thing to work on, the new order may change:

- which entry is examined first,
- the order in which work is scheduled,
- which candidate wins when several are equally good,
- and therefore which of several equally valid results the script arrives at.

6.2 What it does not mean

This is not corruption and it is not randomness. For the same input the runtime behaves the same way every time. The script keeps its complete logical state and execution stays valid.

What changed is a tie-break that was never guaranteed in the first place.

6.3 What to do about it

If a script needs a defined order, it must impose one rather than relying on the order a table happens to produce. Collect the keys and sort them:

SQUIRREL

```
// Fragile: the order depends on the runtime's hash function.
foreach (key, value in prices) {
    consider(key, value)
}

// Defined: the order is the script's own choice.
local keys = prices.keys()
keys.sort()
foreach (key in keys) {
    consider(key, prices[key])
}
```

Both `table.keys()` and `array.sort()` are part of the Squirrel standard delegates available in Simutrans.

Arrays are not affected: they are ordered containers and their order is part of their contract. **Prefer arrays where order carries meaning.**

6.4 Saved script state

When a script's persistent state is written out again, the entries of a saved table may appear in a different order than before. The saved state itself is unaffected: it is restored **by name, not by position**.

7. Migration guide for script authors

7.1 If your script already works

There is no migration step. Concretely:

1. **Test it on a current build.** Run the scenario or AI as you normally would.
2. **Inspect runtime warnings and errors.** No new Squirrel errors were produced by the validation campaign; if your script produces one, it is worth reporting.
3. **Review code that iterates a table to make a decision.** If the first match wins, or the last one does, decide whether that was intentional and make it explicit.
4. **Do not rewrite anything that is not broken.** No API changed.

One caution about testing method: **do not compare two runs by their logs**. Two runs of the same script under different Simutrans versions may make different, equally valid choices. A log diff will show noise that is not a defect.

7.2 If your script fails

Work through this sequence in order. It is designed to separate the causes quickly.

1. **Capture the actual Squirrel error.** The message and the reported line are the starting point; do not diagnose from symptoms alone.
2. **Check the Script API usage against the reference.** The API did not change in this migration, so an API error usually means the script was already using something that had changed at another time, or was never correct. Check the [Script API changelog](#) for the version you targeted.

3. **Check iteration assumptions.** Look for `foreach` over a table where the outcome depends on which entry comes first. This is the one category this migration can plausibly have affected.
4. **Check for behaviour you inherited from the old runtime.** If the script relied on a quirk of the previous vendored snapshot rather than on documented Squirrel behaviour, that is the likeliest remaining cause.
5. **Separate Script API problems from pakset and scenario assumptions.** A missing object, a renamed way or a pakset difference produces failures that look like script failures but are not. Confirm which one you have before changing code.

NOTE

If the failure survives all five steps and is reproducible, it is worth a forum report with the error text, the Simutrans revision, and a minimal script that reproduces it.

8. Scenario authors

Expected compatibility. Scenarios do not need changes. The scenario entry points, the Script API and the savegame format are all unchanged.

Recommended validation.

- Load the scenario on a current build and play far enough to exercise its rules and its win/lose conditions.
- Save and reload mid-scenario, and confirm the persistent state is intact.
- Check any rule that iterates a table of factories, halts, cities or players to pick a target or to award something. If two candidates could be equally good, the winner may now be the other one.

Semantic caveats.

- The iteration-order change affects *which* of several equally valid outcomes occurs, not *whether* the scenario works. A scenario whose text says "the first factory found" may want to say something more specific.
- If a scenario's expected result is asserted against a fixed value that was derived from a table iteration, make that ordering explicit as shown in section 6.3.

9. AI authors

Expected compatibility. The two scripted AI players shipped with Simutrans, `sqai` and `sqai_rail`, were run against the new runtime and remained compatible. In the tested campaign they preserved their logical connection state, continued making valid progress and produced no script or runtime errors. Loading a savegame written by the previous runtime was also tested and works.

Recommended validation.

- Run a campaign of realistic length rather than a short smoke test. AI behaviour differences accumulate over time and a two-minute run will not show them.
- Confirm the AI's persistent state survives a save/load cycle.

- Watch for an AI that stops making progress, rather than for an AI that makes different progress. The second is expected.

Semantic caveats.

- **Scheduling and route-search choices may differ.** This is the expected consequence of the iteration-order change, and it is not a defect.
- An AI that iterates its candidate targets and takes the first acceptable one will now potentially take a different acceptable one. If a particular priority matters, sort the candidates explicitly.
- Because of this, an AI test that asserts on an exact sequence of built connections is testing the runtime's hash function rather than the AI. Assert on properties — the network is connected, the balance is positive, no errors were raised — instead.

10. Paksets

This migration requires no pakset changes of any kind.

Specifically:

- **No pak object-format migration.** The binary pak format is untouched.
- **No `.dat` migration.** No parameter was added, removed or reinterpreted.
- **No graphics migration.**
- **No `menuconf.tab` or configuration change.**

Scripts and pakset assets are separate concerns. A pakset that ships scenarios is affected only through those scenarios, exactly as described in section 8, and only in the iteration-order sense — the pakset content itself needs nothing.

11. Examples

All examples below are valid Squirrel and use only features available in Simutrans at the integration revision.

11.1 Environment binding — both forms

The existing method form is unchanged and continues to work:

```
local env = { factor = 2 }
local scale = function(x) { return x * factor }
local bound = scale.bindenv(env) // 'this' inside scale is now env
bound(21)                       // 42
```

SQUIRREL

Squirrel 3.2 adds an inline form. The environment is written in square brackets between the function keyword and the parameter list:

SQUIRREL

```
local env = { factor = 2 }
local scale = function [env] (x) { return x * factor }
scale(21)                                // 42
```

The same bracket form is accepted for named functions, for local functions, and for class and table members including `constructor`. Both forms produce a closure whose environment is the given table, class or instance.

Portability note. The inline form is a *new language feature*. A script that uses it will not run on older Simutrans versions. The method form runs everywhere.

11.2 `table.map`

SQUIRREL

```
local prices = { coal = 10, oil = 25 }
local labels = prices.map(function(key, value) {
    return key + "=" + value
})
```

The callback is called once per entry and receives the key and the value, with `this` bound to the table. `labels` is an `array`, and its element order follows table iteration order — sort it if that matters.

Portability note. `table.map` is a *new standard-library feature*. A script that uses it will not run on older Simutrans versions.

11.3 Iteration with a defined order

SQUIRREL

```
// Collect the keys, impose an order, then work through them.
local keys = prices.keys()
keys.sort()
foreach (key in keys) {
    do_something(key, prices[key])
}
```

This pattern is worth applying anywhere the choice made inside the loop is visible in the result — the target an AI builds to, the factory a scenario picks, the entry a report lists first.

12. Notes for Simutrans developers: the embedding layer

This section is for people who touch `src/squirrel/` or the Script API registration layer. It is a summary of what moved, not a source walkthrough.

12.1 Instance lookup

`sq_getinstanceup()` gained a fifth parameter, `SQBool throwerror`, which selects whether a type-tag mismatch raises a Squirrel error or returns a plain failure to the caller.

Every call site was updated to preserve its existing behaviour:

- `src/simutrans/script/api_class.h` and `src/simutrans/script/api/api_map_objects.cc` pass `SQTrue`, keeping Simutrans's existing "wrong type is a script error" contract.
- `src/squirrel/sq_extensions.cc` passes `SQTrue`.
- The vendored standard-library files (`sqstdblob.cc`, `sqstdio.cc`, `sqstdstream.cc`, `sqstdstring.cc`) follow upstream, which uses `SQFalse` where a failed lookup is handled locally.

If you add a new Script API class, follow `api_class.h`: pass `SQTrue` so that a type mismatch surfaces to the script author as a Squirrel error rather than as a silent null.

12.2 `bindenv` and closure creation

`SQVM::CLOSURE_OP()` gained a `boundtarget` parameter, and the compiler emits it for the inline `function [env] (...)` form. Nothing in the Script API registration layer depends on this; it matters only if you are modifying the compiler or the VM.

12.3 String hashing

`_hashstr()` in `src/squirrel/squirrel/sqstring.h` was replaced with the upstream 3.2 version. It walks the string in the opposite direction and uses a different step calculation. This is the source of the table iteration-order change described in section 6.

Practical consequence for engine work: anything that iterated a Squirrel table and depended on the resulting order — including test expectations — must impose its own ordering. This applies to C++ code and to the test suite as much as it does to scripts.

12.4 Registration invariants

The migration was gated on the Script API not drifting. The invariants that were held constant, and that should be held constant by future work in this layer:

Invariant	Value	Scope
<code>register_function(vm)</code> call sites	79	<code>src/simutrans/script/</code>
<code>log_squirrel_type</code> call sites	7	<code>src/simutrans/script/api/</code>
Registration drift	0	—
Calling-convention drift	0	—
Publicly exposed type drift	0	—

The scopes are stated explicitly because they are easy to get wrong: counting `register_function(vm)` under `src/simutrans/script/api/` alone yields 77, not 79. Both figures are correct for their own scope; only the wider one is the invariant.

12.5 The preservation principle

The rule this migration followed, and which is worth keeping for any future runtime update: **a runtime update may change how the VM is driven, but it must not change what the Script API exposes**. If a runtime change forces a visible API change, that is a separate, separately reviewed piece of work.

13. Script API stability

The migration was validated for unintended public-API drift rather than assumed to be safe. The checks were run on a fresh checkout of the official trunk at r12168.

Result: no drift of any kind.

- Registered function count: **79**, unchanged from the frozen pre-migration record (scope: `src/simutrans/script/`).
- Type-logging sites: **7**, unchanged (scope: `src/simutrans/script/api/`).
- Registration drift: **0**.
- Calling-convention drift: **0**.
- Publicly exposed type drift: **0**.
- Files changed by the documentation revision that contain code: **0**.

In plain terms: every function the Script API offered before the migration is offered after it, under the same name, taking the same arguments, and exposing the same types.

14. Validation summary

All figures below come from a **fresh checkout of the official trunk at r12168**, built and run after the commits were published — not from the working copy the commits were made from.

284 / 284
REGRESSION

10 / 10
VM TESTS

2 / 2
ORPHAN TESTS

Check	Result
Full regression suite	284 / 284
VM-specific tests	10 / 10 present and passing
Orphan tests (tests that would silently vanish if the suite regressed)	2 / 2 present and passing
New Squirrel errors	0
Script API drift	0 (see section 13)
Documentation build	269 HTML files, 0 new diagnostics against the accepted baseline

Additional validation performed during the migration:

- The two shipped scripted AI players were run in a campaign against the new runtime.

- A savegame written by the previous runtime was loaded repeatedly, to confirm the result was reproducible rather than a single lucky run.
- A state loaded under 3.2 was saved again and reloaded successfully.
- Both intermediate runtime states were verified as independently shippable before anything was published, so no revision in the r12165–r12168 range leaves the tree in a broken state.

The test suite is name-checked rather than count-checked: the VM and orphan tests are verified to be present *by name*, so a suite that silently regressed to an older list would be detected rather than reported as a passing total.

15. Known limitations and notes

15.1 Table iteration order

Documented in [section 6](#). This is the one behavioural difference an existing script may observe, and it is a consequence of an upstream change rather than a Simutrans decision.

15.2 Scope of the compatibility statement

The compatibility statement covers the scripted AI players shipped with Simutrans and the savegame directions listed in section 5. **Third-party scripts were not tested.**

15.3 Backward direction not claimed

Loading a savegame written under Squirrel 3.2 into an older Simutrans build was not part of this work and is not claimed to work.

15.4 New features are not backward compatible

The inline `bindenv` form and `table.map` are new. A script that uses either will not run on older Simutrans versions. If a script must run on both, use `bindenv` as a method and avoid `table.map`.

15.5 Building the Script API documentation

This note is relevant only if you build the Script API reference yourself with Doxygen.

`documentation/DoxyfileSQAPI.in` carries `FILTER_PATTERNS = *.cc`, which recent Doxygen versions reject. This is **pre-existing and unrelated to the Squirrel 3.2 runtime** — it was present before this migration and was deliberately left alone rather than fixed inside an unrelated change. It deserves its own small patch.

A second, environmental point for Windows builders: the documentation build must invoke the input filter through a shell wrapper. Without it, the native Windows Doxygen exits successfully while silently dropping the entire API reference. Check the generated file count, not the exit code.

16. Where to find what

This document does not duplicate the Script API reference. Use it as a map.

You want	Go to
API details — classes, methods, parameters, constants	The Script API reference, generated from <code>src/simutrans/script/api/</code> (<code>api_doc.h</code> is its main page)
What the API gained or changed, and when	<code>src/simutrans/script/api/changelog.h</code> → the <i>Changelog</i> page
Migration and compatibility for Squirrel 3.2	This document, and <code>src/simutrans/script/api/squirrel_32_en.h</code> → the <i>Squirrel 3.2 runtime update</i> page, which is its in-source counterpart
The same, in Spanish	<code>src/simutrans/script/api/squirrel_32_es.h</code> → <i>Actualización del runtime de Squirrel 3.2</i>
Common scripting mistakes	The <i>Common pitfalls</i> page in <code>api_doc.h</code>
Available Squirrel standard-library functions	The <i>Available functions from the squirrel standard lib</i> page in <code>api_doc.h</code>
Worked example scripts	The example pages in <code>api_doc.h</code>

The in-source page at `squirrel_32_en.h` is the **canonical technical reference** for this migration. This document is its public, standalone presentation: it adds the audience-specific guidance in sections 7–10, the validation figures in sections 13–14, and the embedding notes in section 12, and it omits nothing material from the in-source page.

The Spanish page in `squirrel_32_es.h` is a **reviewed translation**; the English page remains canonical where the two could ever be read differently.

17. Availability and changelog

Integrated in Simutrans Standard trunk: r12165 – r12168, with r12168 as the completed migration and documentation state.

Available in builds containing r12168 or later. No numbered stable release is claimed here; check the release notes of the build you are using.

Changelog

Revision	Change
r12165	CODE: update Squirrel instance lookup API for 3.2 — adapt every call site to the new <code>sq_getinstanceup()</code> signature, preserving existing behaviour.

Revision	Change
r12166	CODE: update Squirrel bindenv support for 3.2 — carry the bound target through closure creation so the inline environment form compiles and runs.
r12167	CODE: reconcile remaining Squirrel 3.2 runtime changes — the upstream string hash, <code>table.map</code> , and the remaining VM and header differences, plus the runtime licence file.
r12168	DOC: document Squirrel 3.2 runtime migration — the bilingual in-source migration reference and its links from the main page and the changelog.

18. Document history

Version	Date	Change
1.0	2026-08-16	First public release. Prepared from the validated migration at trunk r12168.