Part 6 — Building the screens

← Contents · Previous: Part 5 — Call flow reference · Next: Appendix

How a client app turns the engine's API into screens for the businesses it serves: the flow editor, the calls page and call detail, the reports, and the live screens. Each section names the endpoints to use, what to do, and what not to do. The demo console (`console/`) is a working example of every screen here.

6.0 The shape of a client app

```
browser (business staff)  ──►  client app backend  ──►  Ringmaster engine (/v1, API key)
                                          ▲
                                          └──  webhooks (signed) from the engine
```

Do

- Call the engine **only from the client app's backend**. The browser talks to the client app, never to the engine.
- Map your business to an engine account once (`externalRef` = your business id) and keep the `accountId`.
- Store what webhooks tell you: calls (`call.ended`), voicemails, summaries, call-out results. Your own reports and search then run on your own database, and the engine stays the source for detail (timeline, recording).
- On start-up, call `GET /v1/ping`: it returns the client app's `features`. Show only the screens the app may use (no Call-outs screen without `CALLOUTS`, no AI blocks without `AI`).
- Switch on the error `code`, show the `message` (README). Messages are written for people.

Don't

- Don't put the API key, the webhook secret or a lobby token for someone else in a page, a mobile app or a log.
- Don't poll the engine to find out that something happened. Webhooks tell you (§6.5).
- Don't fail on fields, enum values, timeline events or webhook types you don't know. The engine adds them within `/v1`.
- Don't show engine words to business staff when yours are clearer: "script" is "call flow" in this book; `MISSED` can be "Nobody answered".

6.1 The flow editor

Load

What	Endpoint	Cache
Block library (types, groups, exits, approval)	`GET /v1/scripts/schema`	Per `schemaVersion`; reload when it changes
The account's flows and their versions	`GET /v1/accounts/{id}/scripts` (D1)	—
A version's blocks	`GET …/scripts/{ref}/versions/{v}` (D6)	Versions never change: cache forever
Pickers: departments, audio, knowledge docs, app actions	I2, E2, F2, `GET /v1/actions`	Short
Other flows for Schedule call (OUTGOING) and Transfer (SHARED)	D1, then D6 of the chosen version	—

Draw the blocks

Build the library from the schema's `blocks`, grouped by `group`, with ⚠ on `needsApproval`. Each block's ports (its exits) come from its settings:

Block	Ports
`menu`	each `options[].id` (default: the key) · `noInput` · `notUnderstood` (only when `speech` is on) · `later`
`collect`	`ok` · `failed` · `later`
`poll`	`next` · `noInput` · `later`
`pick`	`chosen` · `empty` · `noInput` · `later`
`ask`	`answered` · `silence`
`connect`	`noAnswer` · each `wr.options[].exit`
`schedule`	`scheduled` · `notUnderstood` (only with `ask`) · `declined` (only with confirm) · `refused`
`request`	`mode: continue` → `next`; else `success` · `failure` · `timeout` · each `wr.options[].exit`
`app_action`	from the registered action (`GET /v1/actions`): no `wait` → `next`; `wait` → each `outcomes[]` · `failed` · `timeout` · each `wr.options[].exit`
`condition`	`test.kind: hours` → `open` · `closed`; others → `yes` · `no`
`goto`	`tooMany` (when `maxLoops` > 0); draw `target` as a dashed line, not a port

Block	Ports
`transfer`	the `exit` of each Return block in the chosen SHARED version
`ai_agent`	`goal` · `human` · `vm` · `cb` · `nohelp` · `time` · `err`
`say` · `optout`	`next`
`voicemail` · `goodbye` · `return`	none: an end block

Show the `later` port as optional ("if the caller says *later*"). Show ports with nothing connected in a different colour: the call ends there.

Keep the layout yourself

The engine stores the flow, not the drawing. A top-level `layout` key is dropped on push. Store block positions in the client app, by account, flow `ref` and block id. Since ids stay the same across versions, the drawing carries over.

Edit

- **Versions never change.** "Edit" means: load the latest version into a **draft** kept by the client app, change it, and push it as a new version. Keep the draft in your database so a closed tab loses nothing.
- Forms per block follow Part 5: every setting there, with its default as the placeholder. Leave a field out instead of sending its default.
- Block ids: generate them (`b1`, `b2`… or a slug of the name) and **never change an existing block's id**. Stats are counted by id.
- Settings that point elsewhere use pickers, never free text: departments, audio, other flows, app actions.
- Variables: offer a picker with what the flow can use at that block (Part 5 §5.3).
- Send request secrets show as `(hidden)`. Leave them as they are to keep them; let the user type a new one to replace.
- Without the `AI` feature, hide the AI Agent block and the `smart` matching choice.

Check, save, publish

```
 every change (debounced ~1 s)  →  POST /v1/scripts/validate {definition}   → show issues on the blocks
 Save                           →  POST …/scripts/{ref}/versions {version: latest+1, runnable, definition}
 Approve (risky versions)       →  POST …/versions/{v}/runnable
 Go live                        →  PUT /v1/numbers/{numberId}/script {scriptVersionId}
```

```
(incoming)
                                    or pick the version in a call-out
(outgoing)
```

- Show `errors` in red on their block, `warnings` in amber; `block: null` issues at the top (Part 5 §5.9). Disable Save while there are errors.
- `version` = the highest existing version + 1. A `409 VERSION_EXISTS` means someone else saved first: reload, show their version, and let the user re-apply their change. A retry of the same push is safe (same document → `200`, not a new version).
- **Approval** is the client app's rule (2B, D7). A common rule: versions with `riskyBlocks` are pushed with `runnable: false` and a manager approves them; the rest go out `runnable: true`.
- Saving never changes what callers hear. Only "Go live" does. Say so on the button.
- **Roll back** = put an older runnable version back on the number. No push needed.
- **Stop** a version that must not run any more: `POST …/revoke` (D8). Numbers playing it refuse calls, so move them first.

Delete

`DELETE …/scripts/{ref}` (D4). A `409 SCRIPT_IN_USE` lists what uses the flow (`numbers`, `callouts`, `scripts`): show the list with links. A flow that has run calls is archived, not erased: its call history stays readable.

Per-block stats on the canvas

`GET …/versions/{v}/stats` (D9), for the version shown. Draw on each block:

- `visits` (how many calls came through);
- **drop-off** = `hangups / visits` (callers who hung up here: a long or confusing prompt);
- **confusion** = `retries / visits` and `noInput` (callers who didn't know what to press);
- Send request blocks: `reqFail`, `reqTimeout`, `reqMsP50`;
- Waiting rooms: `gaveUp`.

Stats appear a few minutes after calls end, and only for that version: compare versions side by side, not mixed.

Try it

Push the version and put it on a test number, or call it with a quick call to your own phone (`POST /v1/calls`, needs `CALLOUTS`). Then open the call's timeline (§6.2): it shows every block, sentence, key and match.

Don't

- Don't edit the JSON of a stored version and push it under the same number.
- Don't let users type department ids, audio ids or flow refs.
- Don't push on every keystroke: check on every change, push on Save.
- Don't put a version on a number before the push returned `201` (or `200` for the same document).

6.2 Calls page and call detail

The calls list

`GET /v1/calls?accountId=…&page=…&size=50` (G1), newest first. Filters: `from` / `to`, `type`, `result`, `status`, `number`, `calloutId`.

Column	From	Show as
When	`startedAt`	Local time of the business; "Today 14:05"
Who	`from` (incoming) / `to` (outgoing)	Your contact's name if you have one, else the number
Line	`to` (incoming) / `from` (outgoing)	The number's name in your app
Type	`type`	Incoming · Call-out · OTP · Agent call
Result	`result`	A coloured label (table below)
Handled by	`handledBy` (+ agent from the timeline)	Flow · Agent Asha · AI · Voicemail
Length	`durations.totalSeconds`	2:31
□ · □	`recorded` · transcript	Icons; play inline

- **Live calls**: `status=LIVE`, or `status` not `ENDED`. Refresh this tab on `call.started` / `call.ended`, not with a timer.
- For search over months (by contact, by tag), use your own copy of `call.ended` events.

Result labels

`result`	Label	Colour
`HANDLED_BY_SCRIPT`	Served by the flow	green
`HANDLED_BY_AGENT`	Talked to an agent	green

`result`	Label	Colour
`HANDLED_BY_AI`	Served by AI	green
`VOICEMAIL`	Left a message	blue
`MISSED`	Nobody answered	red
`ABANDONED`	Hung up while waiting	red
`HUNG_UP_EARLY`	Hung up at once	grey
`REJECTED`	Refused (limits, suspended)	grey
`ANSWERED` (outgoing)	Answered	green
`NO_ANSWER` · `BUSY` · `DECLINED`	Not answered · Busy · Declined	amber
`UNREACHABLE` · `INVALID_NUMBER` · `FAILED` · `NOT_DIALED`	Couldn't reach · Wrong number · Failed · Not called	grey

Call detail

One page per call, from four calls, loaded together:

Part	Endpoint	Notes
Header: who, when, result, lengths, flow version, outcome tags	`GET /v1/calls/{id}` (G2)	`durations` as a bar: waiting · talking · on hold · AI
Call log	`GET /v1/calls/{id}/timeline` (G3)	Below
Recording	`GET /v1/calls/{id}/recording` (G4)	Proxy it through your backend; `404` = none or not ready
Transcript and summary	`GET /v1/calls/{id}/transcript` (G5)	`404` until `summary.ready`; show "Being transcribed..." if `recorded`

`providerCost` is what the call cost to run (carrier, AI). Show it to the client app's own staff, not to the business, unless that is how you bill.

The call log (timeline ? readable lines)

The timeline is the call's log: every step with its time. Show it as lines, `offsetMs` as `0:12`. Group lines under the block they happened in (`block`). Show the readable ones by default; put the rest behind "Show technical details".

Event	Line	Default
`CALL_RECEIVED` / `DIALING` / `ANSWERED`	"Call from +255745... to Mapokezi line" / "Calling..." / "Answered"	shown
`BLOCK_ENTER`	Block heading: the block's name and type	as heading

Event	Line	Default
`SAY`	◫ "Karibu PMK Chemicals." (`data.text`)	shown
`PLAY`	◫ the audio's name	shown
`KEY`	⌨ "Pressed 2" (red if `valid: false`)	shown
`SPEECH` / `AI_HEARD`	◫ "nataka kununua"	shown
`MATCH`	→ "Chose **Mauzo** (by voice, 92%)"	shown
`INPUT`	⌨ "Entered 1234 (valid)" or `••••`	shown
`NO_INPUT` / `NOT_UNDERSTOOD` / `RETRY`	"No answer — asked again (2)"	shown
`POLL` / `PICK`	"Answered 4 (Nzuri)" / "Chose Arusha"	shown
`CONDITION`	"Opening hours: open"	shown
`QUEUED` / `RING` / `ACCEPTED` / `RING_TIMEOUT`	"Waiting for Mapokezi" / "Ringing Asha" / "**Asha** answered" / "Asha didn't answer"	shown
`WAIT_OPTION`	"Pressed 9 while waiting → Voicemail"	shown
`VOICEMAIL`	"Left a message" with the player	shown
`CALL_SCHEDULED`	"Callback booked for tomorrow 09:00"	shown
`ACTION_REQUESTED` / `ACTION_RESULT`	"Asked the app: pay" / "pay → paid"	shown
`REQUEST`	"GET shop.example → 200 in 340 ms"	shown
`AI_SAID` / `AI_STAGE` / `AI_TOOL`	◫ the sentence / stage / tool	shown
`EXIT_UNCONNECTED`	⚠ "The flow ends here (exit *noInput* not connected)"	shown: it's a flow bug the business can fix
`ERROR`	⚠ `data.code` explained	shown
`ENDED`	"Ended: caller hung up" (`endReason`, `endedBy`)	shown
`WORKER_*` · `AI_USAGE` · `WAIT_UPDATE` · `TRANSFER_IN/OUT` · `VARIABLE_MISSING` · others	as they are	hidden

Link each block heading to the flow editor at that block, on the version the call ran (`script.version` from G2).

Don't

- Don't send the recording's URL with the API key to the browser. Stream it through your backend, after checking that the user may hear it.
- Don't show a sensitive value; the engine already masks it as `••••`, and so should anything you add.
- Don't poll for the transcript: wait for `summary.ready`.

6.3 Reports

Account dashboard

`GET /v1/accounts/{id}/stats?from=…&to=…&groupBy=day` (N1).

Tile / chart	Formula
Calls	`incoming` + `outgoing`
Didn't get a person	`missed` + `abandoned`: callers who waited for an agent and gave up or weren't answered; as a share of `incoming`
Left a message	`voicemail`
Served without a person	(`byScript` + `byAi`) / (`byScript` + `byAgent` + `byAi` + `voicemail`)
Hung up at once	`hungUpEarly` / `incoming` (a long or unclear greeting)
Average call length	`seconds` / `answered` (`answered` counts incoming calls and answered outgoing ones)
Busiest hours	`groupBy=hour`: a heat map by hour
Per line	`groupBy=number`
By type	`groupBy=type`

Range: at most 400 days. Default view: the last 7 days.

Flow performance

- **Funnel per block**: D9 on the canvas (§6.1).
- **Outcome tags** (`countAs`, App action outcomes): per call in `outcomes` (G2, `call.ended`), per call-out in `byOutcome` (H8). For account-wide numbers by tag, count them in your own copy of `call.ended`, or from the CSV export (`outcomes` column).

Call-outs

Screen part	Endpoint
List and status	`GET /v1/callouts?accountId=…` (H5)
Header: status, counts, pause reason	`GET /v1/callouts/{id}` (H7)
Progress bar, by result, by try, by hour	`GET /v1/callouts/{id}/stats` (H8)

Screen part	Endpoint
People and their result	`GET /v1/callouts/{id}/recipients?status=…` (H6)
Download	`GET /v1/callouts/{id}/export.csv` (H9)

While it runs, update the page from `callout.progress` (every 30 s) instead of polling. Show `callout.paused` reasons as actions: `VERSION_REVOKED` (choose another version), `FEATURE_OFF` (ask the owner), `ACCOUNT_SUSPENDED` / `CLIENT_SUSPENDED` (resumes when lifted), `CAN_CALL:<reason>` (your own can-call check said no). Quiet hours don't pause a call-out: it waits and goes on by itself.

Agents

`GET /v1/accounts/{id}/agents/stats` (I13): per agent `offered`, `answered`, `missed`, `declined`, `talkSeconds`. Answer rate = `answered` / `offered`. Live status comes from `agent.status`.

Exports

`GET /v1/calls/export.csv?accountId=…&from=…&to=…` (G6) and H9. Stream them through your backend as downloads; up to 100,000 rows, oldest first.

Don't

- Don't compute reports by walking `GET /v1/calls` page by page. Use the stats endpoints, the exports, or your own copy of `call.ended`.
- Don't mix time zones: stats days and hours are in the **account's** time zone, timestamps are UTC.

6.4 Live screens

The agent desk

Everything is in Part 2D and the desk signals in Part 3. In short:

1. Agent signs in to the client app → backend `PUT …/agents/{ref}` (once) and `POST …/agents/{ref}/lobby-token`.
2. The page joins the LiveKit lobby room with that token and listens for `ring`, `ring_cancel`, `connecting`.
3. **Start shift** button: unlocks the browser's sound (a ring needs a user gesture) and sets status `AVAILABLE`.
4. On `ring`: show the caller and the flow's notes, ring, flash the title. Accept → backend `POST /v1/calls/{id}/accept` → join the call room with the returned token. Decline → `POST`

`…/decline`.
5. In the call: Hold, Transfer (to a department), Hang up, each through the backend.
6. Renew the lobby token before it expires (600 s).

Queue wall

`GET …/departments/{id}/queue` (I6) for the waiting callers. Refresh it when `agent.status`, `call.started` or `call.ended` arrive for that account, with a slow timer as backup (≥ 10 s).

Don't

- Don't let the browser call `accept` or `hangup` on the engine directly. The backend checks the agent is who they say.
- Don't ring with an `<audio>` element created before the user clicked anything: browsers block it.

6.5 Which webhook updates which screen

Webhook	Update
`call.started` / `call.answered`	Live calls list; queue wall
`call.block`	The caller's answers as they happen (a menu choice, a number, a poll answer); act on them in your app (open a ticket)
`call.ended`	Calls list, call detail, your own call store (reports, search)
`call.missed`	"Missed calls" inbox; notify the business
`call.scheduled`	Scheduled callbacks list
`voicemail.received` · `recording.ready`	Voicemail inbox; the call's player
`summary.ready`	Transcript and summary on the call; voicemail preview
`ai.unanswered`	"Questions the AI couldn't answer": suggestions for the knowledge base
`action.requested`	Not a screen: do the action and answer (2E)
`request.failed`	An alert for the business's integration
`callout.progress` · `callout.paused` · `callout.completed` · `callout.recipient_results`	Call-out page; your own record of who was reached
`agent.status`	Agent list, desk, queue wall
`media.ready` · `media.failed`	Audio library (processing → ready)

Webhooks can arrive twice or out of order: store by `id`, and compare `createdAt` before overwriting newer data.

6.6 Don't, in one place

1. Never put the API key or webhook secret in a browser, app or log.
2. Never edit a stored version: push a new one.
3. Never change an existing block's id: stats and links are by id.
4. Never let users type ids (departments, audio, flows, actions): use pickers.
5. Don't poll for what a webhook tells you (calls, recordings, summaries, call-out progress, agent status).
6. Don't fail on unknown fields, values, events or webhook types.
7. Don't offer what the client app's features don't allow (`GET /v1/ping` → `features`).
8. Don't put a version live that is waiting for approval; don't approve in the same click as saving a risky version.
9. Don't show `••••` values unmasked, or engine costs to businesses unless you bill by them.
10. Don't build reports by paging through `GET /v1/calls`.

Revision #1
Created 13 November 2026 18:54:28 by Admin Qbit
Updated 13 November 2026 18:54:50 by Admin Qbit