AUTHCHA LEARN

Guest and portals

Vouchers, captive portals and gateways: let visitors online without a shared Wi-Fi password.

5 modules · about 135 minutes · Network and front-desk administrators

1. Guest vouchers
2. Captive portals: what Authcha does and does not do
3. Build a pfSense captive portal
4. Self-service guest and employee portals
5. Phones, laptops and the sign-in window

Quizzes, the final assessment and live links are online at learn.authcha.cloud. Labs use the live demo at demo.authcha.cloud.

Module 1 of 5 · 30 minutes

Guest vouchers

By the end you can

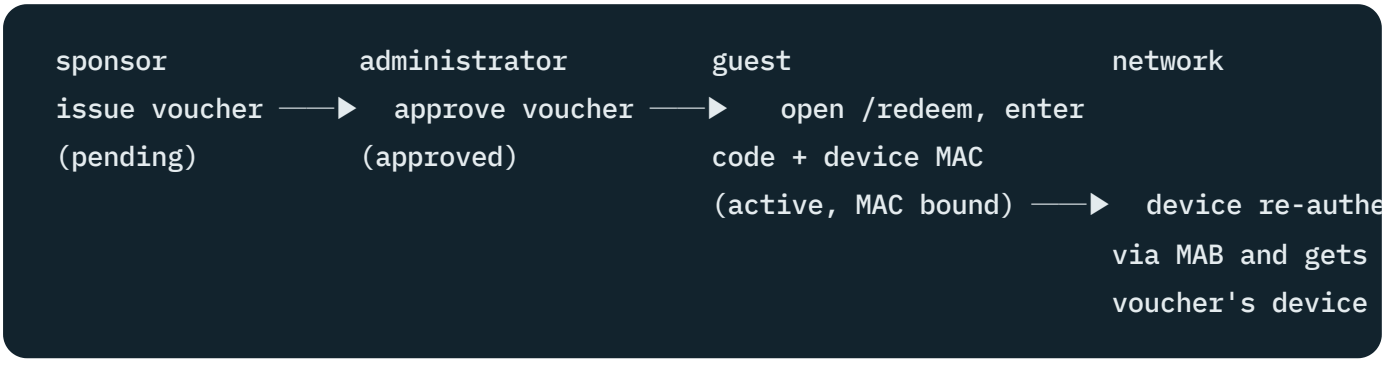
- Describe the four steps from issuing a voucher to a device being admitted
- Prepare the network so a guest device lands in the right place
- Issue, approve, renew and revoke a voucher, and explain each status
- Explain why a failed redemption always looks the same

The idea

Visitors need network access for a few hours or days, without a shared Wi-Fi password that everyone eventually knows. Authcha uses **vouchers**:

1. A **sponsor** (say, front-desk staff) issues a code.
2. An **administrator** approves it.
3. The **guest** redeems the code on a public web page.
4. The guest's device is admitted automatically.

You need the **Guest Access** module in your license edition, and **Network Access Control** for the underlying MAC admission.



When a guest redeems, their device's MAC is added to the MAC allowlist with the voucher's device class (default `guest`), and from then on the device is admitted through the normal MAB path you met in the Network track. When the voucher expires or is revoked, the entry is disabled and the device is refused at its next authentication.

> **Separation of duties**
>
> Having one person issue and another approve gives you a clear audit trail. An administrator may do both, and both steps are still recorded. Sponsors cannot approve their own vouchers.

Roles

Role	Can do
Sponsor	Issue vouchers and see *their own*. Sees only the Dashboard, Vouchers and their Profile
Admin	Everything a sponsor can, plus approve, revoke and renew any voucher and see every voucher
Viewer	No access to vouchers. A voucher code is a usable credential, so the read-only role never sees it

Create sponsor accounts under **Administrators** → **Accounts** → **Add admin** with the **sponsor** role.

Prepare the network

1. **The guest class.** Under **Device Classes**, give `guest` the VLAN, QoS, session limit and bandwidth guests should get. Add vendor attributes if your controller needs them.
2. **Guest access points or switches.** Register them under **Devices**, add them as RADIUS clients and turn **MAB on**.
3. **Pre-authentication access.** Configure the guest SSID or port so a device not yet on the allowlist lands in a restricted state where it reaches only DHCP, DNS and Authcha's HTTPS address (a walled garden or pre-auth ACL). Then redirect its web traffic to the redemption page. The feature name depends on your vendor: external captive portal, web redirect or MAB-failure fallback.
4. **Email (optional).** Configure **Integrations** → **Email (SMTP)** so approved vouchers can be emailed to the guest.

The redemption page

Guests open `https://<authcha-host>/redeem`. It needs no login and asks for the **voucher code** and the **device MAC address**. Your captive portal can fill both in with URL parameters, for example `/redeem?mac=aa:bb:cc:dd:ee:ff&code=GY7OEVOUG6BH`. After success the device must authenticate again to get guest access, so the guest reconnects or the controller re-runs MAB.

Why every failure looks the same

For security, a failed attempt gives the **same answer whatever the reason** (unknown code, expired, already used, revoked, bad MAC), so the page cannot be used to guess codes. Each failure also takes a random 100 to 300 milliseconds so timing does not leak anything, and attempts are limited to 10 per minute per address. The guest sees a generic message and a random **support code**. If a guest needs help, they quote that code to you, and you find the real reason (`bad_code`, `expired`, `revoked`, `exhausted` and so on) under **Vouchers → Analytics → Redemption attempts**.

> **Do not try to be helpful in your own portal**
>
> If you build a portal that calls the redeem API, show the generic message and the support code. Do not try to distinguish failures in the portal, or you rebuild the code-guessing hole.

Issue, approve, renew, revoke

Go to **Vouchers → Issue voucher**.

Field	Meaning
Issued to	Guest name or reference
Email	Optional. The code is emailed when the voucher is approved
Valid until	Expiry time, at most 180 days ahead
Max redemptions	How many different devices can use the code. Default 1
Device class	Policy for the guest's devices. Default `guest`
Allowed subnets	Optional. Limit the voucher to devices on segments you name

Status	Meaning
Pending	Issued, waiting for approval. Cannot be redeemed
Approved	Ready to redeem
Active	Redeemed; one or more devices bound
Expired	Past its end time; its devices are no longer admitted
Revoked	Cancelled by an admin; its devices are refused immediately

Administrators can **renew** (extend, up to 180 days from now), and sponsors and administrators can **resend** the email. **Revoking always works**, whatever your license status.

Expiry is enforced twice

A background sweep every 20 seconds marks expired vouchers, disables their allowlist entries and ends their live sessions. Independently, every MAC authentication checks the voucher's end time, so a stopped sweep can never extend access. When a voucher ends, Authcha sends a Disconnect-Request for the device's open sessions. Devices that support CoA end the session at once. Ones that do not, such as pfSense, have it queued for the portal agent.

Limit a voucher to certain segments

With **allowed subnets**, a device using the voucher is admitted only through network devices assigned one of those subnets. A network device with **no subnet assigned fails closed**: guests using a scoped voucher on it are refused.

Lab HANDS-ON

1. Open Vouchers (https://demo.authcha.cloud/vouchers) on the demo as `demo-admin` (password `Authcha-Demo-2026`). Look at vouchers in different statuses.
2. Open **Analytics** and read the redemption rate and the failure reasons.

3. Issue a voucher for "Test visitor" valid for one day. Note it is **pending**.

4. Approve it. Note the status change.

5. Open `/redeem` on the demo host and try your code with a made-up MAC address. Then try a wrong code. Compare what you see.

6. Revoke the voucher and watch its status.

7. On your own install, do the same, then check **Access Control → Disconnect log** after revoking a voucher with an active guest.

Cheat sheet KEEP THIS

- Flow: **sponsor issues, admin approves, guest redeems at `/redeem`, device admitted by MAB** in the voucher's class (default `guest`).
- Status: **Pending, Approved, Active, Expired, Revoked**. Max validity **180 days**.
- Needs: guest class with a VLAN, MAB on the guest devices, pre-auth walled garden to Authcha's HTTPS.
- Redeem failures are all identical. Support code to **Analytics → Redemption attempts**.
- Expiry enforced by a 20 s sweep **and** at every authentication.
- Scoped vouchers fail closed on devices with no subnet.
- Read more: Guest access (https://demo.authcha.cloud/docs/guest-access).

Module 2 of 5 · 20 minutes

Captive portals: what Authcha does and does not do

By the end you can

- Say which jobs belong to the gateway and which belong to Authcha
- List what Authcha will never do on a gateway
- Pick the right integration path for a given gateway
- Explain what "emulated" means in the compatibility table

The two halves

A **captive portal** is a gateway (a pfSense firewall, a wireless controller, a hotspot router) that holds a new client on a restricted network, shows a login page, and lets the client through once it is authenticated.

The most important sentence in this track: **the gateway intercepts and admits. Authcha is the RADIUS server and policy brain it asks.** For pfSense and MikroTik hotspots, Authcha also writes the login page and hosts the helper pages for vouchers, verification codes and status.

What Authcha does

Job	How
Decides who may log in	The gateway sends credentials to Authcha as a RADIUS Access-Request (PAP). Authcha accepts or rejects
Tells the gateway what the person gets	Reply attributes: VLAN, session limits and the web policy name
Keeps the session record	The gateway's RADIUS accounting is stored. **Sessions** shows who is online
Ends sessions	An RFC 5176 Disconnect-Request to a gateway that accepts it; the router's REST API for a MikroTik with REST settings; a queue the agent drains for pfSense CE
Issues guest access	Vouchers and the self-service guest and employee portals
Writes the login page (pfSense, MikroTik)	**Portal gateways** generates the branded, translated, accessible page you upload
Hosts helper pages	Voucher login, the verification-code stage for OTP user types, status and sign-out, and an RFC 8908 Captive Portal API address

What Authcha does not do

- It does **not intercept or redirect** a client's traffic. The gateway must capture the first web request and send the browser to a login page.
- It does **not admit a client on the gateway by itself**. The gateway authenticates its own clients against RADIUS. No Authcha page tells the gateway "let this address through", and Authcha holds no gateway credential, except an optional REST user you give for a MikroTik.
- It does **not reach into a gateway** except by RADIUS, an optional reachability test of the portal address, and that MikroTik REST call.
- It does **not filter web traffic**. That is web policy, covered in the Integrations track.

A design choice, not a gap

For pfSense, the username and password go **straight from the browser to pfSense**, never through an Authcha page. Authcha deliberately builds no relay for passwords, because only the firewall can admit the client and a relay would be one more place for a password to leak.

Choose the path

Gateway	How it connects
pfSense CE	RADIUS PAP; Authcha generates the page pfSense serves. Vouchers use MAC login; codes use a single-use token. The next module walks through it
MikroTik RouterOS hotspot	The same, with the hotspot's own page files generated by Authcha; Disconnect on UDP 3799 or the REST API
Any NAS that redirects and accepts CoA (Cisco, Aruba, Ruckus, HPE, Juniper, Extreme, Fortinet, generic)	**Central Web Auth**: Authcha hosts the sign-in page and tells the NAS to let the client in by CoA. Nothing is uploaded to the NAS
Wi-Fi controllers (UniFi, Meraki, Aruba, Cisco WLC, FortiGate, and modules for Ruckus, Omada, Mist)	The controller redirects to Authcha as an external portal, and its API admits the client
Open-source hotspots (CoovaChilli, nodogsplash, openNDS and others)	Authcha's landing page and the gateway's own admit mechanism
WiFiDog	Not supported

Be honest about test status

The guides say plainly how far each integration was tested. **Emulated** means the flows were exercised against a mock written from the gateway's documented protocol, not against real firmware. Today the pfSense portal was verified in a lab on pfSense 2.9.0 for login, accounting, the agent's sign-out and the generated pages, and the others are emulated or mock-tested. Every guide has a "not verified" list. Read it

before you promise a customer a specific gateway, and plan a proof of concept on their real equipment.

> **Tell customers what has and has not run on real gear**
>
> A guide that says "mock-tested only" means exactly that. Do not turn it into "supported on firmware X" without testing.

Lab HANDS-ON

1. Open Portal Gateways (https://demo.authcha.cloud/portal-gateways) on the demo (viewer login `demo-viewer`, password `Authcha-View-2026`). See the gateways, their zones and settings.
2. Open the Captive portals guide (https://demo.authcha.cloud/docs/captive-portal) and find the compatibility matrix. For three gateways of your choice, write down the status.
3. Open the Captive portals on Wi-Fi controllers guide (https://demo.authcha.cloud/docs/captive-portal-controllers) and read its "not verified" section.
4. For a gateway at your own site, decide which row of the table applies and what you would need to test.

Cheat sheet KEEP THIS

- **Gateway intercepts and admits. Authcha decides and tells.**
- Authcha does not intercept traffic, admit clients on a gateway itself, or filter web traffic.
- Paths: pfSense (page + PAP), MikroTik (hotspot files), **CWA** (any NAS with redirect + CoA), controllers (external portal + API), open-source hotspots.
- "Emulated" or "mock-tested" is **not** real-gear tested. Plan a proof of concept.

- Read more: Captive portals (https://demo.authcha.cloud/docs/captive-portal), Central Web Auth (https://demo.authcha.cloud/docs/captive-portal-cwa).

Module 3 of 5 · 40 minutes

Build a pfSense captive portal

By the end you can

- Register the firewall in Authcha and create the portal zone on pfSense
- Set the RADIUS, accounting and HTTPS options that make sessions work
- Generate the branded pages and upload them
- Diagnose the most common portal failures

What you get

When you finish, visitors who join the guest network and open any web page see a branded login page, in English, Hindi, Spanish, French or German, that works on phones and desktops, in light and dark. Behind it:

- Username and password go **straight from the browser to pfSense**, which checks them with Authcha over RADIUS.
- A **Use a voucher** button opens an Authcha page that registers the device, and pfSense's MAC login then admits it.
- User types you choose (for example `human-write`) sign in with password **and** a verification code, then get a single-use token that admits them.
- A status page shows who is signed in, time online, data left and an **End my session** button.
- Error cases (wrong password, locked account, quota used up, expired password, unregistered device, too many tries) each get a clear message in the visitor's language.

The integration was written against pfSense 2.7 and verified in a lab on **2.9.0**; other versions are unverified. You need Authcha 1.14 or later with a license that includes guest access (the trial does), and the console reachable from the guest network by a **name whose certificate the guests' devices trust**, because the helper pages are HTTPS-only.

1. Register the firewall in Authcha

1. **Authorization Policies** → **RADIUS Clients** → **Add**: the address the firewall's RADIUS packets come from, and a shared secret.

2. **Devices** → **Add**: the same address, device type **Captive portal / hotspot (PAP)**, and **CoA mode: none** (pfSense CE has no listener). If pfSense puts a different address in `NAS-IP-Address`, add it under **Additional addresses**. For vouchers, turn on **MAC authentication bypass** with the **pfSense MAC auth** preset.
3. **Users**: create users of type `network-user` for people who need only network access. Optionally add a network policy and a web policy.

2. Create the zone on pfSense

Services → **Captive Portal** → **Add**. Give a zone name (lowercase letters, digits, underscore, such as `guest`), tick **Enable Captive Portal**, choose the interface (such as the guest VLAN) and save.

3. Authentication and accounting

On the zone's configuration page:

pfSense setting	Value
Authentication method	**RADIUS Authentication**
RADIUS protocol	**PAP**
Primary RADIUS server	the Authcha host, port `1812`
Shared secret	the one from step 1
Reauthenticate users every minute	**Off** (a code login's password is a single-use token)
Accounting	Enable, port `1813`, same secret
Accounting updates	**Interim update**, about every 60 seconds
Use RADIUS reply attributes	On (timeouts, bandwidth, quota)
Enable RADIUS MAC authentication	On if guests use vouchers; MAC format single dash or colon

pfSense sends the **client's IP address** as Calling-Station-Id. Authcha keys sessions on that, so sessions, the status page and the one-person rule all work by address.

4. HTTPS and the walled garden

- Enable **HTTPS login** and set **HTTPS server name** to the name guests reach, with a certificate their devices trust. Without it every browser warns before the login page. The verification-code stage **needs** HTTPS login.
- **Allowed hostnames**: add the **Authcha console host name** in both directions, and its IP under **Allowed IP addresses**, on port 443. Without this the voucher, code, status and logo pages cannot load.
- **Firewall rule**: on the portal interface, pass TCP 443 from portal clients to the console, above any block rule.

5. Create the gateway in Authcha

Guest Access → **Portal Gateways** → **New gateway**. The important fields:

Field	Value
Name, zone	any name; the zone from step 2
Portal login address	`https://<firewall>:8003/index.php?zone=<zone>`
Device	the device from step 1
Visitors can	username and password, or also a voucher
Verification code for	the user types that must pass a code (empty = none)
Original page after sign-in	`any`, `allowlist` or `none` (never an open redirect)
Look and text	brand name, accent colour (Authcha checks the contrast), a logo up to 20 KB, welcome, terms and help text
One person per device address	On, unless the console is reached through NAT or many people share an address

Save, then open the gateway and choose **Set up the firewall**.

6. Upload the pages

Download the two files and paste them into the zone:

File	pfSense field
`authcha-portal.html`	**Portal page contents**
`authcha-portal-error.html`	**Authentication error page contents**

Each file is checked by an offline validator as it is generated, and the console will not let you download one that fails. The page carries a Content-Security-Policy with hashes of its own style and script, so **editing the HTML by hand breaks it on purpose**. Change settings in Authcha and download again. **Test the connection** checks that the console can reach the portal address and that the page is Authcha's, and reports problems in plain words (name not found, connection refused, timed out, a certificate problem).

7. Try it

1. Connect a phone to the guest network and open any `http://` page. The branded page appears.
2. A wrong password shows a message in the visitor's language. A right one lands on the page they asked for.
3. **Sessions** and **Portal gateways** → **Sessions** show the session once pfSense's accounting Start arrives.
4. **End session** removes it: Authcha queues the disconnect and the pfSense agent ends it.

> **Without the agent, sign-out does nothing on pfSense CE**
>
> pfSense CE has no CoA listener, so ending a session only **queues** the disconnect. The **pfSense agent** collects and applies it. Without the agent the user, and the next person on a shared machine, stay connected.

When it goes wrong

Symptom	Cause and fix
The page is pfSense's default	The file is not in **Portal page contents**. Test the connection
Voucher, code or status page will not open	The console is not in the walled garden (name **and** IP), or its certificate is not trusted
Certificate warning before the login page	The portal's HTTPS server name has no trusted certificate
"Authentication failed" for an OTP user	By design from this gateway: use **Sign in with a verification code**, or the three-tries lock (**Users** → **Unlock**)
Logged in, then thrown out after a minute	**Reauthenticate users every minute** is on with the code stage
Voucher accepted but device not admitted	MAC authentication is off on the zone, or the MAB mode does not match, or the MAC format differs
"No active session" while online	Requests reach the console from the firewall's NAT address, not the client's
Every login says the RADIUS server did not answer	Wrong source address or secret, or no Message-Authenticator. **Auth History** shows nothing for a dropped request

Lab HANDS-ON

1. Open Portal Gateways (https://demo.authcha.cloud/portal-gateways) on the demo (viewer login `demo-viewer`, password `Authcha-View-2026`) and look at a gateway's settings.
2. Read the pfSense settings table above and tick off, from memory, the six pfSense options that matter most.
3. On a pfSense lab (2.7 or later), follow steps 1 to 7 end to end.

4	Break step 4 on purpose: remove the console from the walled garden. See which pages fail, then fix it.		
5	Turn on	**Reauthenticate users every minute**	and watch a code login fail after a minute. Turn it off again.
6	End a session from	**Portal gateways → Sessions**	with the agent stopped, then with it running. Compare.

Cheat sheet KEEP THIS

1. RADIUS client + device (**Captive portal / hotspot (PAP)**, **CoA none**, MAC auth preset).
2. Zone on pfSense.
3. **RADIUS PAP**, 1812/1813, interim updates, **reauth off**, reply attributes on.
4. **HTTPS login** + walled garden (name **and** IP) + TCP 443 rule.
5. **Portal Gateways → New gateway** in Authcha.
6. Upload **two files** (portal and error pages). Never hand-edit them.
7. Test with a phone. The **agent** ends sessions on pfSense CE.
8. Read more: pfSense / generic captive portal (https://demo.authcha.cloud/docs/pfsense-captive-portal), phones and laptops (https://demo.authcha.cloud/docs/captive-portal-devices).

Module 4 of 5 · 25 minutes

Self-service guest and employee portals

By the end you can

- Explain the guest registration flow and the two approval modes
- Configure the settings that limit who can register
- Explain how employees register devices and certificates
- Give employees the right advice about private Wi-Fi addresses

Two pages that save the front desk

Two self-service pages let people get onto the network without anyone issuing anything by hand. Both are **off** until you enable them under **Guest Access** → **Portals** → **Settings**.

- **Guest registration** at `/guest`: visitors register with their name and email, confirm the address, and get a voucher code.
- **Employee devices** at `/byod`: employees sign in with their work account, register their own phones and laptops, and can create their own certificate for 802.1X.

Guest registration

1. The visitor opens `https://<your-server>/guest`. Your captive portal can send them there with `?mac=aa:bb:cc:dd:ee:ff` to fill in the device for the last step.
2. They enter name and email, accept the terms and choose **Send me a link**. The link works **once** and expires after one hour.
3. When they open it, in **automatic** mode they get a voucher at once (also emailed), and in **sponsor** mode they are told someone will approve it. A sponsor or admin approves under **Portals** → **Guest requests**, and the code is emailed.
4. The guest chooses **Connect this device**, which opens the redeem page with the code filled in.

Because the result is an ordinary voucher, everything from the vouchers module still holds: the class decides the VLAN and limits, the voucher expires by itself, and you can revoke it.

Setting	Meaning
Approval	Automatic after the email check, or a sponsor decides
Device class	Where guests are placed
Access lasts, devices per guest	Validity (up to 30 days) and number of devices (up to 5)
Only these email domains	Restrict registration, for example to a partner's staff
Never these email domains	Block throwaway-mail providers
Notify	Who is emailed when a request is waiting (sponsor mode)
Terms	Text guests must accept

Guest registration needs working email under **Settings** → **Integrations**. Without it the page reports that it is unavailable, because it cannot confirm addresses.

Abuse protection

The page never reveals whether an address is known. Each address and each source address is rate-limited. The confirmation link is random, single-use and stored only as a hash. One request per address is live at a time. A hidden field catches simple bots. If you expect abuse, use **sponsor mode** or restrict the allowed domains.

Employee devices

1. The employee opens `https://<your-server>/byod` and chooses **Sign in**, using Authcha's own sign-in page: passkey, or password and one-time code. Locked accounts and disabled user types cannot sign in.
2. They add a device by its Wi-Fi MAC address, with an optional name. It is admitted by MAC bypass in the class you chose and listed under **Network Access** → **Endpoints** with the owner's name.
3. If you allow certificates, they can create an EAP-TLS certificate: a `.p12` file and a one-time password. **The private key is not stored.** They can revoke their own certificates.

Setting	Meaning
Device class	The class registered devices get
Devices per person	Up to 20
Registration expires after	Optional. Expired devices are switched off within a minute
Who can use it	Limit to specific user types
Certificates	Allow self-service certificates, with lifetime, limit and class

Everything is scoped to the signed-in person: nobody can see or remove another person's devices or certificates, and a device an administrator already registered, or another person owns, cannot be claimed.

> **The private Wi-Fi address trap**
>
> Tell employees to turn off **private Wi-Fi address** (iOS, Android and Windows call it different things) for your network **before** registering a device. A randomised MAC changes, and the registration stops matching. For laptops, 802.1X with a certificate avoids the problem completely.

Activity and limits

Portals → **Activity** lists registrations, approvals, devices and certificates for 180 days. In this release guests cannot be sent a code by text message (email only), and the employee portal registers devices by MAC address but does not provision Wi-Fi profiles onto the device. The public demo cannot send email, so it shows the confirmation link on the page and turns certificates off.

Lab HANDS-ON

1. Open Portals (https://demo.authcha.cloud/portals) on the demo as `demo-admin` (password `Authcha-Demo-2026`) and look at the settings and activity.

2 Open `/guest` on the demo host and register with a test address. Because the demo cannot send email, use the confirmation link it shows.

3 Approve the request under **Portals → Guest requests** if the demo is in sponsor mode, and note the voucher that results.

4 Open `/byod` and read the sign-in page. (On your install, sign in and register a phone's MAC.)

5 On your install, restrict guest registration to one email domain and test that another domain is refused.

Cheat sheet KEEP THIS

- `/guest` (visitors) and `/byod` (employees). Both **off** until enabled under **Guest Access → Portals → Settings**.
- Guest: name + email → single-use link (1 h) → voucher (automatic or sponsor-approved) → **Connect this device**.
- Limits: guest voucher up to **30 days**, **5** devices; employee **20** devices.
- Abuse protection: no address reveal, rate limits, hashed single-use links, one live request per address, bot trap.
- Employees: turn off **private Wi-Fi address**, or use 802.1X with a certificate.
- Read more: Portals (https://demo.authcha.cloud/docs/portals).

Module 5 of 5 · 20 minutes

Phones, laptops and the sign-in window

By the end you can

- Describe how a device discovers a captive portal and when its sign-in window closes
- Name the probe addresses the gateway must handle
- Explain the design rules that keep the portal working in a sign-in mini-browser
- Predict the effect of a private Wi-Fi address on vouchers and MAB

How a device meets the portal

1. The device joins the Wi-Fi and asks a **probe address**.
2. The gateway intercepts the probe and answers with a redirect to the portal.
3. The operating system shows its **sign-in window**, a mini-browser, or a notification. The person signs in and the portal sends the browser on.
4. The gateway lets the client through. The next probe gets its genuine answer, and the system closes the window and marks the network online.

Steps 2 and 4 are the **gateway's** job. Authcha's part is the page in step 3 and the helper pages it links to.

The probes

The gateway must intercept these before sign-in and answer genuinely after:

Probe	Genuine answer
`captive.apple.com/hotspot-detect.html`	`Success` page
`connectivitycheck.gstatic.com/generate_204` and similar	HTTP 204, no body
`www.msftconnecttest.com/connecttest.txt`	`Microsoft Connect Test`
`detectportal.firefox.com/success.txt`	`success`
`nmcheck.gnome.org/check_network_status.txt`	`NetworkManager is online`

pfSense's captive portal does this itself. Authcha **does not answer probes**, because they go to other hosts. It only has to avoid breaking them. If a phone keeps saying "no internet" after a good login, check the zone's allowed hostnames and that the gateway passes the probe through once the client is admitted.

The operating systems

System	How the window appears	What to remember
iOS and macOS	The Captive Network Assistant mini-browser opens when the probe is not answered `Success`	No pop-ups, cookies may be sandboxed, no downloads. Private Wi-Fi Address is **on by default**
Android	A "Sign in to Wi-Fi network" notification opens a WebView	Old WebViews (Android 7) understand only old JavaScript. Randomised MAC per network is on by default from Android 10
Windows 10 and 11	The system opens the default browser or an Edge-based window	A normal browser. HTTPS to the portal needs a trusted certificate or it warns
Linux and ChromeOS	A notification offers to open a browser	A normal browser

Why the pages are built the way they are

The sign-in windows are limited browsers. The generated pages use nothing they are known to lack:

Rule	Why
No pop-ups, no new windows	The sign-in windows cannot open them, so every step stays in the same view
No cookies and no browser storage	Storage may be empty or blocked. Form protection is a signed, address-bound token in the form
No file downloads	Mini-browsers cannot download
No outside requests	Before sign-in the client reaches only the walled garden, so style, script and the logo are inline
Works without JavaScript	Every action is a plain form post or link. Script only adds friendliness
Old-browser JavaScript only	iOS 12 Safari and Android 7 WebView must parse it
Relative units and system fonts	Text scaling to 200% and right-to-left languages work without a sideways scrollbar

> **What was and was not tested**
>
> The guide is clear: the pages were checked against engines, screen sizes and user agents, but **the real sign-in assistants were not run**. Plan a test on actual phones before a launch.

Native "Sign in" with RFC 8908

Authcha can answer the **Captive Portal API** at `https://<console>/p/<gateway>/capport`. Advertise it with **DHCP option 114** (and the Router Advertisement option for IPv6), allow the console in the walled garden and use a trusted certificate. Phones that implement it then show a native "Sign in" button and the time and data left. It is optional, and the redirect flow keeps working without it. It was **not verified on a device**.

Private Wi-Fi addresses

iOS, Android and Windows use a different MAC address per network, and some rotate it. The consequences:

- **Sessions** are keyed by client IP, so a rotating MAC does not confuse the status page or session control.
- **Vouchers are tied to the MAC the device presented when it redeemed.** If the address changes, the device is a new one: the portal asks for the voucher again, and the voucher's device count includes the old address. An administrator can revoke it and issue a new one.
- **MAB entries** for known devices need a stable address. Turn private addressing off for that network on managed devices, or use 802.1X.

The only place a changing address costs anything is the per-voucher device limit.

Lab HANDS-ON

1. Open the phones and laptops guide (https://demo.authcha.cloud/docs/captive-portal-devices) and find the "Not verified" list for your operating systems.
2. On a phone, connect to a captive network (the demo has none, so use your lab pfSense or a hotel network) and watch the sign-in window open and close.
3. Check **Private Wi-Fi Address** on an iPhone or **Wi-Fi privacy** on an Android phone for your lab network. Redeem a voucher, then change the setting and reconnect. What happens?
4. On a Linux laptop in the guest network, run `curl -i http://detectportal.firefox.com/success.txt` before and after sign-in. Compare the answers.
5. List the five probe addresses from memory.

Cheat sheet KEEP THIS

- Flow: **probe, redirect, sign-in window, admit, genuine probe answer, window closes.**

- Gateway handles probes. **Authcha does not.**
- Portal pages: no pop-ups, no cookies or storage, no downloads, no outside requests, work without JavaScript.
- **Private Wi-Fi address** breaks voucher and MAB matching. Turn it off, or use 802.1X.
- RFC 8908: `https://<console>/p/<gateway>/capport` and **DHCP option 114** (optional, unverified on a device).
- Read more: Captive portal: phones, laptops and assistants (https://demo.authcha.cloud/docs/captive-portal-devices).