Obsah

Port 8080 — HTTP/HTML webserver **1**
Parametry spojení . . . 1
Zpracování požadavku (NHttpSocket::MsgReady) . . . 1
Relace a autentizace (NSession) . . . 2
Oprávnění podle úrovně přihlášení (od 2.15.beta12) . . . 2
GET endpointy . . . 3
Události zásahů (NEvents) . . . 3
Stránky (HTML) . . . 3
JSON data (AJAX) . . . 3
Diagnostika . . . 5
Statické soubory . . . 5
Konfigurace lokací a zařízení . . . 5
Ruční ovládání a testy . . . 6
Povely regulace (ruční, z webu) . . . 6
Systém . . . 7
POST endpointy . . . 7
Šablony . . . 7
Pozn. k nasazení . . . 7

Port 8080 — HTTP/HTML webserver

Platí pro verzi NPA R04: 2.13.beta20 · Vytvořeno: 2026-06-02

> Zdroj: `server.cpp` / `server.h` (třída `Server`, singleton `Server::GetServer()`), `nhttpsocket.{h,cpp}` (parsování požadavku), `nhtml.{h,cpp}` (skládání odpovědí), `nsession.{h,cpp}` (relace/oprávnění).

Konfigurační a monitorovací webové rozhraní jednotky. Surový `QTcpServer` (ne `QHttpServer`) s ručním parsováním HTTP.

Parametry spojení

Parametr	Hodnota	Zdroj
Naslouchá na	`0.0.0.0`	pevně
TCP port	**8080**	pevně (`Server::Init`)
Socket	`NHttpSocket` (potomek `QTcpSocket`)	

Příchozí spojení (`incomingConnection`) připojí slot `readClient()` na `readyRead`.

Zpracování požadavku (`NHttpSocket::MsgReady`)

- Požadavek se akumuluje do `mMsg`, zpracuje se po ≥ 3 řádcích.
- `mUrl` = první řádek (percent-decoded), tj. `GET /a/b/<command>?<param> HTTP/1.1`.
- **Příkaz** = 4. segment cesty: `commands = path.split("/")`, `command = commands[3]`. Parametry za ? jsou v `pagess[1]`, vícenásobné odděleny &.
- `mSession` = hodnota cookie `NsessionId` (jinak se vygeneruje nové UUID).
- `mEAUser` = hodnota `ea_role` z cookie (externí autentizace).
- `mPost` = tělo požadavku (dle `Content-Length`, percent-decoded, + → mezera).

Relace a autentizace (NSession)

- Stav drží NSession (mUserName, mAuthLevel, mEditLocationUuid, mEditDeviceUuid, mIoLocationUuid …); web jej vyzvedne přes mNHtml.SetSession(mSession, mEAUser).
- Cookie NsessionId se nastavuje v každé HTML hlavičce (HtmlHead): Set-Cookie: NsessionId=<id>; Path=/.
- Přihlášení: POST login.html → NSession::Login(post); odhlášení: logout.html; změna hesla: POST password.
- Typy odpovědí: HtmlHead (text/html), JsonHead (application/json), JsHead, CssHead.

Oprávnění podle úrovně přihlášení (od 2.15.beta12)

Každý příkaz (GET i POST) se před zpracováním kontroluje proti minimální úrovni (server.cpp → AuthRank / CommandMinRank). Dříve se stránky jen skrývaly v menu (menu<úroveň>.html) a přímým zadáním URL byly dostupné všem. Úrovně: "" (nepřihlášen) < user < admin < guru < sasa (cookie ea_role z cloudu: admin → guru, za_firmu/technik → admin, user → user).

Cookie ea_role se **přijme jen z VPN tunelu** (IsFromVpn, util/nutils.cpp: adresa klienta v síti rozhraní tun_npa_r04, nebo přímo VPN server x.y.0.1); z LAN se ignoruje a zapíše do logu (cookie ea_role=… od <ip> mimo VPN - ignorovana), takže lokálně ji nelze podstrčit. Relace přihlášená takto vzdáleně (NSession::mRemote) nemá v hlavičce tlačítko **odhlásit** (odhlášení řídí cloud); lokálně přihlášený uživatel ho má.

Minimální úroveň	Příkazy
bez přihlášení	index.html, index2.html, login.html, logout.html, lang, public.html, update_gaude, ping, getid, statické soubory (style.css, npa.css, fonts.css, main.js, main1.js, gauge.min.js)
user	basic_settings.html, POST save, POST password, get_index_data, get_bin, get_da, get_device_detail, device_detail.html, getstate, get_history, tok_energie.html, get_tok_energie
guru	info.html, config.html, data.html, state.html, log.html, debug.html, instr, clear_fram, settestbatt
admin	**vše ostatní** (nastavení, zařízení, rozpadová místa, povely regulace, restart/reboot, IEC 104, Modbus registry, zkoušky dle RfG, přesun VPN, get_diag); neznámý příkaz = admin (bezpečný default – nová stránka pro uživatele se musí přidat do seznamu)

Odpověď při nedostatečné úrovni: stránka (*.html) → html/v2/forbidden.html s menu (nepřihlášený dostane přihlašovací stránku), akce / JSON → {"res":"error","msg":"bez opravneni"}; pokus se zapíše do logu (HTTP GET/POST <prikaz>: bez opravneni).

GET endpointy

Události zásahů (NEvents)

Ruční zásahy z webu se od 2.15.beta12 evidují jako události (zdroj web, kdo = uživatel@IP): všechny povely regulace (set*), rfg_start/stop, testy výstupů (dd_test_out, dd_xch_out, outtogle, releclick, tranzistorclick, rs485click), instr, clear_fram, vpn_switch_start/cancel, iec104_toggle/profil, setda a uložení nastavení (save, lo_save, dd_save, dd_ramps_save, iec104_save, rfg_save, restart_save; u POST jen název příkazu, ne obsah). Zapisují se do /log/npa_events.log (jen změny, opakování se počítá), jdou v poli events paketu /measuring a Modbus část je vidět na stránce Modbus registry. Celý soubor: GET events_log (text/plain, admin+). Vzor pro další zdroj (IEC 104 povely): NEvents::GetNEvents()->Add("<src>", kdo, co, hodnota).

Stránky (HTML)

index.html, index2.html, info.html, net_settings.html, basic_settings.html, factory_settings.html, settings.html, regulation.html, locations_dis.html, devices_detection.html, io_settings.html, param.html, config.html, data.html, state.html, log.html, debug.html, public.html, modbus_registers.html, rfg_tests.html (zkoušky dle RfG, admin+), tok_energie.html (tok energie a průchod regulací, user+), reboot.html, restart.html, login.html, logout.html

JSON data (AJAX)

Endpoint	Obsah
get_index_data	hlavní přehled (NState::GetInxeDataJson)
get_device_detail?<uuid>	detail zařízení (NHtml::DeviceDetailJson): info = HTML text ovladače (Device::GetInfo), state, rows = všechny klíče NState::M(uuid) jako {k, d (popis), u (jednotka), v (hodnota formátovaná serverem), g (meas / diag = debug-* a pomocné klíče energií), min, min_t, max, max_t (z klíče_timed, jen když existuje), reg (registr zDevice::Meta– u Save* automaticky FC/adresa/typ/násobitel), n (vysvětlivka)}, inv = tabulka měničů DataHubu {cols, colreg, rows: [{id: port-adresa, v: [...]}]}; stejný JSON je při načtení stránky vložen do <script type="application/json" id="dd_init">
get_bin?<uuid>	binární I/O lokace (GetInOutJson)

Endpoint	Obsah
get_tok_energie	záložka Tok energie (NHtml::TokEnergieJson, obnova po 2 s, úroveň user): sit (P TML + fáze, elektroměry TML, povolený přetok, SoP), objekt (vlastní spotřeba = P TML + Σ P zařízení DS), mista (P, povel, souhrn řídicích funkcí ctrl jako get_location_ctrl číselně, zařízení místa: P, akt_proc, cil_proc před rampou, potvrzeny_proc, p_deficit, SOC, graf = existuje historie), bez_mista, regulace = snímek řetězce regulace z posledního ticku (NRegulation2::TokJson: distribuce, kandidáti povoleného přetoku s vítězem, požadavky pomalé/rychlé větve, výstup PI, výrobní strop, strop distribuce + kompenzace, skupiny a místa, BESS, cos φ, omezuje = co právě omezuje; pi jen admin+). Hodnoty ve W / %, texty skládá web
get_location_ctrl?<uuid>	editace rozpadového místa (admin): blok Obchodní flexibilita a tabulka Řídicí funkce činného výkonu místa (NHtml::LocationCtrlJson) – lo_extplc_val, extplc_prc, extplc_prc_state, extplc_kw, extplc_kw_state, sum_reg, sum_px, sum_px_label, sum_extplc, sum_maxout, sum_result a *_cls (sum-win = vítězná, nejnižší hodnota); klíče odpovídají tokenům šablony, stránka je obnovuje po 2 s
get_da	stav D/A výstupů (GetDaJson)

Endpoint	Obsah
`get_history?uuid=<uuid|TML|VYR>&range=<900|3600|43200|86400>` nebo `&from=<ms>&to=<ms>`	historie P/Q/energií a povelu zařízení, výrobního místa, souhrnu TML nebo souhrnu výroby VYR (kladná P zařízení DS, energie míst typu výroby) pro graf ve webu (`NHistory::ToJson`). Okno v čase: `range` = posledních N s do teď, nebo `from`/`to` v UTC ms (posun a zoom grafu; server ořízne na `to` ≤ teď, `from` ≥ teď − 24 h, délka 60 s … 24 h a vrátí `from`, `to`, `range` = délka [s]): `step` `raw` (jen okno ≤ 1 h pokryté surovým kruhem, `p = [[t_ms, P_W, Q_VAr|null, povel_%|null, povel_BESS_%|null, P_set_W|null, P_vyr_W|null]]`, každé měření, max 2000 bodů) nebo `1min` (`p = [[t_ms, P_avg, Q_avg|null, P_min, P_max, povel_avg_%|null, povel_BESS_avg_%|null, P_set_avg|null, P_vyr_avg|null]]`), `q15` = čtvrthodiny zarovnané na UTC `[[t0_ms, t1_ms, P_z_energií|null, P_průměr|null, P_vyr_z_energií|null, P_vyr_průměr|null]]`, `nom_w` = jmenovitý výkon (`dd_Power`, u místa součet) pro přepočet povelu na W, `odber_kladny` = TML (+ = odběr, P z energií = (ΔE_con − ΔE_del)/Δt), `has_proc` / `has_procb` = povel měniče (`akt_proc`) / BESS (`akt_proc_bess`, + vybíjení) v rozsahu existuje; jen u TML `has_pset` / `P_set` = nastavený výkon TML [W, + = odběr] = −povolený přetok z regulace (`regulace/pretok_povoleny`, při regulaci po fázích limit na fázi; null = přetoky se neregulují) a `has_vyr` / `P_vyr*` = souhrn výroby přibalený z klíče VYR (stejné časy, vlastní spotřeba = P TML + výroba se počítá v grafu); bez dat `res error`
`getstate`	stav (pole `p_inv`/`q_inv`)
`getid`	sériové číslo + příznak `-unregistered`
`update_gaude`	data pro měřidla (gauge)
`info.html` (JSON větev)	`info` + `test` HTML bloky

Diagnostika

`get_diag` (admin+, text/plain Markdown): verze, info, config bez hesel, aktuální data, uložený stav, **historie výkonu a energií** (`NHistory`: per zařízení / místo / TML CSV bloky čtvrthodiny, minutové agregáty 24 h a celý surový kruh 8192 vzorků – řádově MB) a posledních 30 kB obou logů. `get_diag?file` = stažení jako soubor `npa_diag_<SN>_<UTC>.md`, `get_diag?hist=0` = bez historie (používá tlačítko kopírování do schránky na stránce Info).

Statické soubory

`style.css`, `npa.css`, `fonts.css`, `main.js`, `main1.js`, `gauge.min.js` (servírováno z adresáře `html/`).

Konfigurace lokací a zařízení

Endpoint	Akce
`lo_add` / `lo_edit?<uuid>` / `lo_delete?<uuid>`	přidat / editovat / smazat lokaci
`lo_delete_device?<uuid>`	odebrat zařízení z lokace
`dd_add` / `dd_add_tcp` / `dd_add_485`	přidat zařízení (manuálně / TCP / RS-485)
`dd_edit?<uuid>` / `dd_delete`	editace / smazání zařízení
`setting_next`	přidat prázdné zařízení do seznamu
`io_select_location`	výběr lokace pro I/O
`re_add_accu`	přidat akumulátor

Ruční ovládání a testy

Endpoint	Akce
`releclick?<true/false>`	sepnout/rozepnout relé (`NIOPins::SetRele`)
`tranzistorclick?<true/false>`	tranzistorový výstup (`NIOPins::SetTranzistor`)
`rs485click?<true/false>`	připojit RS-485 linku (`NRs485::SetConnect`)
`setda?<n>&<value>`	nastavit D/A výstup n (test, 30 cyklů)
`settestbatt?<%>`	testovací SOC baterie
`outtogle?<key>&<bit>`	přepnout binární výstup
`dd_test_out?<uuid>` / `dd_xch_out`	test výstupů zařízení
`instr?<uuid>&<fn>&<var>&<addr>&<param>`	vložit Modbus instrukci do fronty driveru
`clear_fram`	smazat perzistentní stav (`mSavedM`)
`rfg_status` / `rfg_start?seq=<1..3>` / `rfg_stop`	zkoušky dle RfG – stav / start sekvence (jen režim SW) / stop; admin+, odpověď JSON; POST `rfg_save` = uložení nastavení (viz `zkousky-rfg.md`)

Povely regulace (ruční, z webu)

Endpoint	Funkce `NState`	Úroveň
`setglobalpower?<W>`	`SetGlobalRegPower`	globální
`setgloballevelp?<n>`	`SetGlobalRegLevel` (stupeň P)	globální
`setgloballevelq?<n>`	`SetGlobalQRegLevel`	globální
`setgloballevelfi?<n>`	`SetGlobalFiRegLevel` (cosφ)	globální
`setglobalbesspower?<W>`	`SetGlobalRegPowerBess`	globální BESS
`setglobalbesslevelch?<n>`	`SetGlobalRegBessLevelCharge`	globální BESS nabíjení
`setglobalbesslevеldisch?<n>`	`SetGlobalRegBessLevelDischarge`	globální BESS vybíjení
`setlocationpower?<W>` / `setlocationlevelp?<n>`	`SetLocationRegPower/Level`	lokace
`setlocbesspower` / `setlocbesslevelch` / `setlocbesslevеldisch`	`SetLocUuidReg(…)`	lokace BESS
`setdevpower?<W>` / `setdevlevelp?<n>`	`SetDevRegPower/Level`	zařízení
`setdevbesspower` / `setdevbesslevelch` / `setdevbesslevеldisch`	(zakomentováno)	zařízení BESS

Globální/lokační povely volají `NState::Save()` (perzistence do `state.cnf`). Lokační/zařízenové povely míří na `mEditLocationUuid` / `mEditDeviceUuid` z aktuální relace.

Systém

lang?<id> (přepnutí jazyka), password, save, restart_save, ping, reboot.html, restart.html.

POST endpointy

Tělo (mPost) se předává příslušné funkci; konfigurační POST zapisují do NConfig a volají NConfig::Save() → /data/NPA_R04.cnf.

Endpoint	Akce
login.html	NSession::Login(post) → html/back.html
save	Save(post) (uložení formuláře konfigurace)
password	změna hesla (NHtml::BasicSettings(post))
restart_save	Save + NConfig::SaveNetConfig → restart
lo_save	NConfig::SaveLocation(uuid, …)
lo_add_device	NConfig::LocationAddDeviceTo(uuid, …)
dd_add / dd_add_tcp / dd_add_485	Save + Devices::AddDevice[/Tcp/485]
dd_save	Devices::SaveDevice(uuid, …)

Šablony

Odpovědi skládá NHtml ze souborů v html/<lang>/ (cz/, en/, plus v2/). Jazyk dle NConfig::mLanguage.GetLang
Statika a sdílené šablony jsou přímo v html/.

Pozn. k nasazení

systemctl/install.sh kopíruje html/ vedle binárky do /usr/local/bin/npa/, takže relativní cesty html/... fungují z pracovního adresáře služby.