Contents

0.1 KingKiosk MQTT Element Architecture Reference

This document describes the KingDSP-style MQTT architecture for KingKiosk.

Canonical control is split into:

1. **System-level commands** (`system/*`) - control the device as a whole
2. **Element-level commands** (`element/*`) - control an individual element (typically a window tile)

Legacy topic families (including `widget/{id}/*`, `.../command`, and any window-scoped command topics) are intentionally **not supported** in the current implementation.

0.1.1 Admin UI Contract (Definitive)

If you are building or rewriting the King Admin interface, treat the following as the stable MQTT API contract:

- **Ingress (commands)**
  - Device/system: `kingkiosk/{device_id}/system/cmd`
  - Element-scoped (optional, only when a widget registers a handler): `kingkiosk/{device_id}/eleme`
- **Egress (responses)**
  - System responses: `kingkiosk/{device_id}/system/response`
  - Element responses: `kingkiosk/{device_id}/element/{element_id}/response`

- **State & events (for UI rendering)**
  - Device capabilities: `kingkiosk/{device_id}/info` (retained)
  - Element state: `kingkiosk/{device_id}/element/{element_id}/state` (retained)
  - Element events: `kingkiosk/{device_id}/element/{element_id}/event` (non-retained)

Rules: - Do **not** publish/subscribe to legacy topic families (no `.../command`, no `widget/...`, no window-scoped MQTT topics). - For any command, if `response_topic` is omitted, the app will default to the canonical system/element response topic based on the ingress topic. - For "full coverage" control surfaces (create windows, move/resize, tiling, etc.), rely on the **System Commands** section (system/cmd). Treat element commands as widget-specific enhancements.

0.1.2 Table of Contents

0.1.3 Overview

The architecture provides two levels of MQTT control:

1. **System-level commands** - control the device as a whole (screen, tiling, etc.)
2. **Element-level commands** - direct control of an individual element/window

0.1.3.1 Key Benefits

- **Granular control**: Send commands directly to a specific element

- **Automatic state publishing**: Registered elements can publish their state automatically
- **Event streaming**: Real-time events from registered elements (errors, state changes)
- **Device discovery**: Retained info topic for capabilities discovery
- **Signed envelopes**: Optional HMAC signing for secure command delivery

0.1.3.2 Implementation Status (Current)

- The app subscribes to canonical element topics. **Only widgets/controllers that explicitly register** will receive element-level commands via the registration-based router.
  - Registration is done via `MqttWidgetMixin` (implementation detail).
  - Unregistered elements are handled via the unified dispatcher (window/tiling/etc.) when applicable.
- `kingkiosk/{device_id}/system/cmd` is accepted by the app and is routed to the unified dispatcher when no explicit system handler is registered. Calendar bidirectional sync details: see docs/CALENDAR_MQTT_SYNC.md.

0.1.4 Feature Server Autodiscovery

KingKiosk clients support automatic discovery of the Feature Server (KingKiosk Core3) via MQTT. This eliminates the need for manual server URL configuration across multiple devices.

0.1.4.1 Discovery Topic

Topic: `kingkiosk/core3/api` (retained)

Payload Format:

```
{
  "api_url": "http://192.168.1.100:3000",
  "version": "3.2.1"
}
```

Fields: - `api_url` (string, required): Full HTTP/HTTPS URL to the Feature Server API endpoint - `version` (string, optional): Server version string for compatibility checking

0.1.4.2 Client Behavior

When a KingKiosk client receives a message on `kingkiosk/core3/api`:

1. **Parse and validate** the `api_url` field
2. **Normalize** the host (extract base URL from api_url if needed)
3. **Check manual override** - If user has enabled "manual override lock", ignore the autodiscovered value
4. **Update server URL** - If not locked, automatically update the Feature Server connection to use the new URL
5. **Connect** - Attempt connection to the autodiscovered server

0.1.4.3 Manual Override Lock

Users can enable "manual override lock" in settings to prevent autodiscovery from changing their manually configured server URL. This is useful for: - Testing against a specific server instance - Using a non-production server - Temporarily isolating a device from the main server

When manual override is enabled, the client: - Still subscribes to `kingkiosk/core3/api` - Logs incoming discovery messages (for debugging) - **Does NOT** update the server URL or reconnect

0.1.4.4 Server Implementation

The Feature Server (KingKiosk Core3) should: 1. **Publish on startup** to `kingkiosk/core3/api` with `retain: true` 2. **Include full URL** in `api_url` (e.g., `http://192.168.1.100:3000`) 3. **Re-publish on config change** if the server URL changes

0.1.4.5 Example: Server Announcement

```
mosquitto_pub -h broker.local -t "kingkiosk/core3/api" -r -m '{
  "api_url": "http://192.168.1.100:3000",
  "version": "3.2.1"
}'
```

0.1.5 Native OS Widgets

KingKiosk supports creating native home screen widgets (Android) and home/lock screen widgets (iOS) that update via MQTT independently of the main app. This is achieved by adding the `os_widget: true` parameter to supported widget creation commands.

0.1.5.1 Supported Widget Types

Widget Kind	iOS Support	Android Support	Notes
`gauge`	⊠	⊠	Radial, linear, and thermometer styles
`chart`	⊠	⊠	Line charts with sparkline view
`weather`	⊠	⊠	Current conditions + forecast
`alarmo`	⊠	⊠	Security system status and controls
`mqttButton`	⊠ (iOS 17+)	⊠	Toggle buttons with state feedback
`sensor`	⊠	⊠	Simple numeric value display

Widget Kind	iOS Support	Android Support	Notes
`counter`	☒	☒	Numeric counter display
`clock`	☒	☒	Current time (no MQTT needed)
`canvas`	☒	☒	Snapshot-based visual diagrams

0.1.5.2 Architecture

Data Flow:

```
MQTT Broker
    ↓
┌────────────────────────────────────┐
│  KingKiosk App (Flutter)           │
│  • MqttOsWidgetMixin               │ ← Intercepts widget creation
│  • WidgetDataService               │ ← Writes to shared storage
│  • WidgetConfig serialization      │
└────────────────────────────────────┘
    ↓ (App Group shared storage)
┌────────────────────────────────────┐
│  Native Widget Extension           │
│  • iOS: WidgetKit                  │
│  • Android: Glance/AppWidget       │
│  • LightMQTTClient (WebSocket)    │ ← Connects to MQTT independently
│  • Reads WidgetConfig from shared  │
│  • Subscribes to configured topic  │
│  • Renders natively                │
└────────────────────────────────────┘
```

Storage Structure: - **Widget configs:** Stored in shared storage at key `kk_widget_{widgetId}` as JSON - **Cached values:** Stored at key `kk_cache_{widgetId}` with value + history - **MQTT config:**

Stored at key `kk_mqtt_config` with broker connection info - **Registered IDs:** List stored at key `kk_registered_widget_ids`

0.1.5.3 Creating an OS Widget

Add the `os_widget: true` parameter to any supported widget creation command. The widget will be created both in-app (optional) and registered as a native OS widget.

0.1.5.3.1 Example: Gauge Widget

Topic: `kingkiosk/{device_id}/system/cmd`

Payload:

```json
{
  "command": "create_gauge",
  "window_id": "temp_sensor_1",
  "gauge_type": "radial",
  "title": "Living Room",
  "min": 50,
  "max": 90,
  "unit": "°F",
  "mqtt_topic": "homeassistant/sensor/living_room_temp/state",
  "json_field": "temperature",
  "os_widget": true,
  "thresholds": [
    {"value": 70, "color": "#4CAF50"},
    {"value": 75, "color": "#FFC107"},
    {"value": 80, "color": "#F44336"}
  ]
}
```

What Happens: 1. `MqttOsWidgetMixin` intercepts the command 2. Extracts a `WidgetConfig` from the payload 3. Calls `WidgetDataService.registerOsWidget(config)` 4. Widget config is written to shared storage as JSON 5. Native widget extension reads the config on its next timeline refresh 6. Widget appears in the device's widget picker (user adds it to home screen) 7. Widget independently subscribes to the MQTT topic and updates

0.1.5.3.2 Example: MQTT Button Widget

Topic: `kingkiosk/{device_id}/system/cmd`

Payload:

```json
{
  "command": "mqtt_button",
  "action": "configure",
  "window_id": "porch_light_btn",
  "mode": "toggle",
  "label": "Porch Light",
  "publish_topic": "zigbee2mqtt/porch_light/set",
  "publish_payload": "{\"state\": \"TOGGLE\"}",
  "subscribe_topic": "zigbee2mqtt/porch_light",
  "status_path": "state",
  "icon": "light_bulb",
  "icon_off": "light_bulb_outline",
  "color_on": "0xFFFFC107",
  "color_off": "0xFF757575",
  "os_widget": true
}
```

Interaction Flow: 1. User adds button widget to home screen 2. Widget displays current state by subscribing to `zigbee2mqtt/porch_light` 3. User taps button on home screen 4. Widget publishes to `zigbee2mqtt/porch_light/set` with payload 5. Widget receives updated state from subscription topic 6. Button color/icon updates to reflect new state

0.1.5.3.3 Example: Alarmo Security Widget

Topic: `kingkiosk/{device_id}/system/cmd`

Payload:

```
{
  "command": "alarmo_widget",
  "window_id": "home_security",
  "mqtt_base_topic": "alarmo",
  "available_modes": ["armed_away", "armed_home", "armed_night", "disarmed"],
  "require_code": true,
  "code_length": 4,
  "os_widget": true
}
```

0.1.5.4 Widget Update Mechanisms

Native OS widgets update via two mechanisms:

1. **Direct MQTT subscription** (primary)
    - Widget extension connects to MQTT broker using `LightMQTTClient`
    - Subscribes to the topic specified in `WidgetConfig.mqttTopic`
    - Updates immediately when messages arrive
    - Frequency limited by OS (iOS: ~15-60min, Android: configurable)
2. **Cached value fallback** (secondary)

- Main app writes latest values to shared storage via `WidgetDataService.writeCachedValue()`
- Widget reads cached value on timeline refresh
- Provides instant display even if MQTT connection fails
- Maintains 48-point history for sparkline charts

0.1.5.5 Widget Lifecycle

Registration:

```
// In MQTT command handler (after creating Flutter widget):
maybeCreateOsWidget(payload, WidgetKind.gauge);
```

Updates:

```
// When widget value changes:
maybeUpdateOsWidgetValue(windowId, value, stringValue: "72°F");
```

Removal:

```
// When widget is closed:
maybeRemoveOsWidget(windowId);
```

Or via MQTT:

```
{
  "command": "close_window",
  "window_id": "temp_sensor_1"
}
```

0.1.5.6 MQTT Config for Widget Extensions

Widget extensions require MQTT connection credentials to operate independently. These are written to shared storage via `WidgetDataService.writeMqttConfig()` when the main app connects to MQTT.

Config Structure (stored at `kk_mqtt_config`):

```json
{
  "wsUrl": "wss://broker.local:8884/mqtt",
  "host": "broker.local",
  "port": 1883,
  "username": "kingkiosk",
  "password": "secret",
  "useTLS": true,
  "allowSelfSigned": true,
  "hmacEnabled": false,
  "hmacSecret": "",
  "deviceName": "kitchen_tablet"
}
```

Important: Widget extensions use **WebSocket** MQTT connections, not TCP: - Secure: `wss://` on port 8884 (not 8883) - Insecure: `ws://` on port 1884 (not 1883)

0.1.5.7 Platform-Specific Implementation

0.1.5.7.1 iOS (WidgetKit)

- **App Group:** `group.com.ki.kingkiosk`
- **Widget Kinds:** `KingKioskGaugeWidget`, `KingKioskChartWidget`, etc.
- **Refresh Policy:** Timeline-based, OS-controlled (15-60 min typical)
- **Interactive Widgets:** Supported on iOS 17+ via App Intents
- **Storage:** UserDefaults with app group suite

0.1.5.7.2 Android (Glance/AppWidget)

- **Widget Receivers:** `GaugeWidgetReceiver`, `ChartWidgetReceiver`, etc.
- **Package:** `com.ki.king_kiosk.widgets.receivers`
- **Refresh Policy:** Configurable update intervals
- **Interactive Widgets:** Full click handler support
- **Storage:** SharedPreferences with process name

0.1.5.8 Debugging

Check registered widgets:

```dart
final service = Get.find<WidgetDataService>();
print(service.registeredWidgetIds); // Set<String>
```

Verify widget config in shared storage:

```
final configJson = await HomeWidget.getWidgetData<String>('kk_widget_temp_sensor_1');
final config = WidgetConfig.fromJsonString(configJson);
print(config.toJson());
```

Check cached value:

```
final cacheJson = await HomeWidget.getWidgetData<String>('kk_cache_temp_sensor_1');
final cache = jsonDecode(cacheJson);
print(cache['currentValue']); // Latest value
print(cache['dataPoints']); // History (up to 48 points)
```

0.1.6 Topic Structure

0.1.6.1 Command Topics (Subscribe)

Topic	Description
`kingkiosk/{device_id}/system/cmd`	New system-level commands
`kingkiosk/{device_id}/element/{element_id}/cmd`	Canonical per-element commands

0.1.6.2 State/Event Topics (Publish)

Topic	Retained	Description
`kingkiosk/{device_id}/info`	Yes	Device capabilities and active widgets
`kingkiosk/{device_id}/status`	Yes	Online/offline status (LWT)
`kingkiosk/{device_id}/system/state`	Yes	System state (tiling mode, screen info)
`kingkiosk/{device_id}/feature_server/state`	Yes	Feature Server connection/settings state snapshot (enabled, connected, reconnecting, URL, errors).
`kingkiosk/{device_id}/element/{element_id}/state`	Yes	Element state
`kingkiosk/{device_id}/element/{element_id}/event`	No	Element events

Topic	Retained	Description
`kingkiosk/{device_id}/element/{element_id}/response`	No	Command response (if correlation_id provided)
`kingkiosk/{device_id}/system/response`	No	System command response

0.1.7 Element Commands

Send commands directly to a specific element using its `element_id`.

0.1.7.1 Topic Format

```
kingkiosk/{device_id}/element/{element_id}/cmd
```

0.1.7.2 Command Payload Format

```
{
  "command": "command_name",
  "correlation_id": "optional-tracking-id",
  ...additional parameters...
}
```

0.1.7.3 Notes

- Element-scoped commands are delivered only to elements that register a handler with `MqttWidgetRouter.registerWidget(...)`.
- Per-element command schemas are widget-specific. This document treats **system commands** as the stable contract; element commands should be considered optional unless explicitly documented for a given widget.
- Window geometry/stacking is controlled via **system commands** on `kingkiosk/{device_id}/system/c` (e.g. `move_window`, `resize_window`, `set_opacity`, `maximize_window`, etc.).

0.1.7.4 Example: Send command to a clock element

Topic: `kingkiosk/my-device/element/clock-1/cmd`

Payload:

```json
{
  "command": "set_mode",
  "mode": "digital",
  "correlation_id": "req-12345"
}
```

Response (on `kingkiosk/my-device/element/clock-1/response`):

```json
{
  "status": "success",
  "mode": "digital",
  "correlation_id": "req-12345",
  "widget_id": "clock-1"
}
```

0.1.8 Element State

Each registered element automatically publishes its state when: - Element is created/registered - After any command is processed - When `publishState()` is called programmatically

0.1.8.1 Topic Format

```
kingkiosk/{device_id}/element/{element_id}/state
```

0.1.8.2 State Payload Example (Clock Widget)

```json
{
  "type": "clock",
  "element_id": "clock-1",
  "widget_id": "clock-1",
  "mode": "analog",
  "visible": true,
  "minimized": false,
  "show_numbers": true,
  "show_second_hand": true,
  "theme": "auto",
  "background_mode": "transparent",
  "background_opacity": 0.6
}
```

0.1.9 Element Events

Registered elements publish non-retained events for real-time notifications.

0.1.9.1 Topic Format

`kingkiosk/{device_id}/element/{element_id}/event`

0.1.9.2 Event Types

Event	Description	Additional Fields
`created`	Widget was created	`type`
`closed`	Widget was closed	`type`
`error`	Error occurred	`message`, `code` (optional)
`state_changed`	State transition	`from`, `to`
`clicked`	User interaction	`x`, `y` (optional)
`ended`	Playback ended	-

0.1.9.3 Event Payload Example

```
{
  "event": "error",
  "message": "Stream disconnected",
  "code": "STREAM_TIMEOUT",
  "element_id": "video-1",
  "widget_id": "video-1",
  "timestamp": "2024-12-19T10:30:00.000Z"
}
```

0.1.10 System Commands

System-level commands control the device as a whole.

0.1.10.1 Topic Format

`kingkiosk/{device_id}/system/cmd`

0.1.10.2 Supported System Commands

System commands sent to `kingkiosk/{device_id}/system/cmd` are routed through the unified command dispatcher.

This section lists the **actual** system command strings that are wired up in the current dispatcher. Detailed parameters for the non-window system commands are documented in the code-derived section System (Non-Window) Commands.

Common notes:

- Many handlers support `response_topic` to control where results are published.
- Unless noted otherwise, `response_topic` defaults to `kingkiosk/{device_id}/system/response`.
- **Widget creation commands** can include `os_widget: true` to also register the widget as a native OS widget (home screen widget on Android, home/lock screen widget on iOS). See Native OS Widgets section for details.

Core system command families:

Category	Commands
Volume	`set_volume`, `mute`, `unmute`

Category	Commands
Brightness	`set_brightness`, `get_brightness`, `restore_brightness`, `request_brightness_permission`, `check_brightness_permission`, `resume_kiosk_after_permission`
Notifications	`alert`, `notify`
Halo	`halo_effect`
Screensaver	`screensaver`, `screen_saver`
Settings / FAB lock	`lock_fab`, `unlock_fab`, `lock_settings`, `unlock_settings`
Person detection	`person_detection`
Screenshot	`screenshot`
Cache	`cache`, `cache_control`, `clear_cache`
TTS	`tts`, `speak`, `say`
STT	`stt`, `speech_to_text`, `listen`
Background	`set_background`, `get_background`
Provisioning	`provision`, `get_config`
AI	`ai_agent`, `ai`, `provision_ai_chatbot`, `setup_ai_chatbot`, `configure_ai_chatbot`
Batch / scripting	`batch`, `kill_batch_script`, `batch_status`, `wait`
Screen schedule	`set_screen_schedule`, `list_screen_schedule`, `enable_screen_schedule`, `disable_screen_schedule`, `screen_schedule_status`, `trigger_screen_schedule`
Conflict resolution	`conflict_resolution`

MQTT button	`mqtt_button`, `mqtt_action_status`, `action_status`

0.1.10.3 Example: Create a clock via system command

Topic: `kingkiosk/my-device/system/cmd`

Payload:

```json
{
  "command": "open_clock",
  "window_id": "clock-living-room",
  "name": "Living Room Clock",
  "mode": "analog",
  "show_numbers": true,
  "theme": "dark",
  "x": 100,
  "y": 100,
  "width": 300,
  "height": 300
}
```

0.1.11 Device Info

The device info topic provides discovery information about the device's capabilities and current state.

0.1.11.1 Topic Format

`kingkiosk/{device_id}/info`

0.1.11.2 Info Payload

```json
{
  "device_id": "my-device",
  "version": "2.1.0",
  "platform": "macos",
  "app_start_timestamp": "2024-12-19T10:00:00.000Z",
  "capabilities": {
    "webview": true,
    "video": true,
    "rtsp": true,
    "webrtc": true,
    "audio": true,
    "visualizer": true,
    "tts": true,
    "stt": true,
    "camera": true,
    "microphone": true,
    "facial_recognition": true,
    "person_detection": true,
    "dlna_renderer": true,
    "screen_share": true
  },
  "widget_types": [
    "webview", "video", "audio", "rtsp", "webrtc",
    "image", "mqtt_image", "map", "visualizer", "gauge",
    "line_chart", "bar_chart", "pie_chart", "carousel",
    "clock", "weather", "calendar", "alarmo",
    "mqtt_button", "timer", "dlna_player", "video_call"
  ],
  "active_widgets": ["clock-1"],
  "widget_count": 1,
  "tiling_mode": "floating",
  "hmac_signing": false,
  "timestamp": "2024-12-19T10:30:00.000Z"
}
```

Notes: - `tiling_mode` is currently a placeholder value in the info payload. - `active_widgets` includes only widgets that have registered with the per-widget router. - `timestamp` is the time this info payload was published (it may be refreshed during the app run). - `app_start_timestamp` stays constant for the lifetime of the running app process.

0.1.12 Signed Envelope Format

For secure command delivery, commands can be wrapped in a signed envelope using HMAC-SHA256.

0.1.12.1 Envelope Format

```json
{
  "ts": 1703001234,
  "msg": "{\"command\":\"play\"}",
  "sig": "a1b2c3d4e5f6..."
}
```

0.1.12.2 Signature Computation

```
sig = hex(HMAC-SHA256(secret, "topic\ntimestamp\nmsg"))
```

Where: - `topic` = the full MQTT topic the message is published to - `timestamp` = the `ts` value (Unix seconds) as a string - `msg` = the JSON-encoded message string

Important: - Signed envelopes are **topic-aware**: the signature depends on the full MQTT topic the message is published to. - If `useSignedEnvelopes` is enabled **and** a shared secret is configured, the app **enforces verification** for inbound signed envelopes. - Invalid signatures or invalid/expired timestamps are rejected (the command is ignored). - If signing is disabled or no secret is configured, the app will still unwrap the envelope for compatibility. - The envelope timestamp used by the implementation is **Unix seconds**.

0.1.12.3 Example (Python)

```python
import hmac
import hashlib
import json
import time

def create_signed_envelope(topic, message, secret):
  ts = int(time.time())
  msg = json.dumps(message, separators=(',', ':'))

  sig_data = f"{topic}\n{ts}\n{msg}"
  sig = hmac.new(
    secret.encode(),
    sig_data.encode(),
    hashlib.sha256
  ).hexdigest()

  return {
    "ts": ts,
    "msg": msg,
    "sig": sig
  }
```

0.1.13 Widget Type Reference

This section is **code-derived**: it documents the JSON keys that are actually parsed/used by the current implementation.

There are three relevant command planes:

1. **System commands**: publish to `kingkiosk/{device_id}/system/cmd`. These create windows/tiles and perform global actions.
2. **Element-scoped commands**: publish to `kingkiosk/{device_id}/element/{element_id}/cmd`. Only controllers that register with the element router receive these.

0.1.13.1 Common Window Geometry Keys

Most "create/open" commands accept the following top-level keys:

Key	Type	Notes
`window_id`	string	Optional. If omitted, an ID may be auto-generated by the tile creator.
`title` / `name`	string	Widget title/name (varies by command).
`x`, `y`	number	Optional position in pixels.
`width`, `height`	number	Optional size in pixels.
`opacity`	number	Optional. Defaults to `1.0`.

Many commands also accept:

Key	Type	Notes
`response_topic`	string	Optional. If omitted, the app defaults to `kingkiosk/{device_id}/system/response` for system commands, or `kingkiosk/{device_id}/element/{elemen` for element commands.

0.1.13.2 System (Window/Layout) Commands

These are **system-level** window/layout commands handled by the main dispatcher. Send them on `kingkiosk/{device_id}/system/cmd`.

Unless explicitly stated, these commands use the payload key `command` to select the handler.

0.1.13.2.1 Window Management (`close_window`, `maximize_window`, `minimize_window`, `bring_to_front`, `send_to_back`)

Accepted command strings:

- Close: `close_window`
- Maximize: `maximize_window`
- Minimize: `minimize_window`
- Bring to front: `bring_to_front` (aliases: `bring_front`, `to_front`)
- Send to back: `send_to_back` (aliases: `send_back`, `to_back`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	One of the management commands above.
`window_id`	string	(required)	ID of the tile/window to act on.

Key	Type	Default	Notes
`response_topic`	string	`kingkiosk/{device}/system/response`	Publishes `{success, command, window_id, timestamp}` or `{success:false, error, ...}`.

0.1.13.2.2 Close All Windows (`close_all_windows`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	Must be `close_all_windows`.
`response_topic`	string	`kingkiosk/{device}/system/response`	Publishes `{success:true, closed_count, ...}`.

Notes:

- After closing tiles, it attempts to stop background audio (best-effort; errors are logged but do not fail the command).

0.1.13.2.3 Window Mode (`window_mode`, `set_window_mode`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	`window_mode` or `set_window_mode`.
`mode`	string	(required)	`tiling`/`tile`, `floating`/`float`, or `toggle`.

Notes:

- This handler currently logs only (no success/error payload is published).

0.1.13.2.4 Update Window Geometry (`update_window`, `move_window`, `resize_window`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	`update_window`, `move_window`, or `resize_window` (all route to the same handler).
`window_id`	string	(required)	Target tile/window ID.
`x`, `y`	number	-	If both present, updates position.
`width`, `height`	number	-	If both present, updates size.

Notes:

- If neither a complete `(x,y)` pair nor a complete `(width,height)` pair is provided, the command is ignored (logged as missing parameters).

- Values for x, y, width, and height are interpreted as **physical pixels** and are internally converted to logical pixels by dividing by the device pixel ratio.
- This handler currently logs only (no success/error payload is published).

0.1.13.2.5 Widget Convenience (show_widget, hide_widget)

Top-level keys:

Key	Type	Default	Notes
command	string	(required)	show_widget or hide_widget.
type	string	(required)	For show_widget: clock, weather, calendar, music, photos, finance, fitness, news (others are ignored). For hide_widget: use all to close all tiles, or any string to match by tile name.
style	string	-	Only applied for clock/weather (stored in the created tile config).
ai_enhanced	bool	false	Only applied for clock/weather (stored in config).

Notes:

- These commands currently log only (no success/error payload is published).
- hide_widget closes the most recently created tile whose name contains type (case-insensitive), unless type == all.

0.1.13.2.6 DLNA Player (`dlna_player`, `open_dlna_player`)

This widget reflects and controls the built-in DLNA/UPnP renderer. It now supports audio, video, and images (and will classify content based on DIDL-Lite metadata and/or URI).

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	`dlna_player` or alias `open_dlna_player`.
`name`	string	`DLNA Player`	Tile title.
`window_id`	string	(auto)	If provided, used as tile ID.
`opacity`, `x`, `y`, `width`, `height`	number	-	Optional geometry.
`response_topic`	string	`kingkiosk/{device}/system/response`	Publishes `{success, command:'dlna_player', window_id, name, timestamp}` or error payload.

Note:

- When `window_id` is omitted, the handler publishes a generated ID in the response (`dlna_{timestamp}`), which may not match the actual auto-generated tile ID used by the controller.

0.1.13.2.7 Remote Browser (`create_remote_browser`, `add_remote_browser`)

This widget provides a thin-client browser experience by streaming a server-rendered Chromium browser over WebRTC through the Feature Server. Designed primarily for tvOS (Apple TV) and iOS devices where local browser rendering is limited.

Architecture: - Server runs a Browser Producer Agent (BPA) with Chromium + Puppeteer - Media is routed through the Feature Server SFU pipeline (H.264 video + Opus audio) - Client connects to the Feature Server/Core (SignalingService) via WebSocket - Control input (pointer, keyboard, navigation) is sent via DataChannel (SCTP) - Telemetry (URL changes, load state, stats) is received via DataChannel - Sessions can be created client-side (when `session_id` is omitted) or joined (when `session_id` + join token are provided)

Prerequisites: - Feature Server must be enabled and connected in Settings > Networked Audio - The Feature Server/Core WebSocket must be reachable (typically `ws://<host>:4000/ws`) - To **join an existing session**, `server_url` must include `?token=<consumerJoinToken>` (required for `transport.*`, `consume`, `dataproducer.*`, etc.) - If `session_id` is **omitted**, the client will call `session.create`, receive a join token, reconnect with `?token=...`, and then publish the resolved `session_id` + tokenized `server_url` in widget state - Newly created sessions can stay in `CREATING` briefly while BPA/Chromium starts; clients will retry `session.join` until the session becomes `READY`/`RUNNING` (or time out)

Top-level keys for `create_remote_browser` / `add_remote_browser`:

Key	Type	Default	Notes
`command`	string	(required)	`create_remote_browser` or `add_remote_browser`.
`window_id`	string	(required)	Unique ID for this browser tile.
`name`	string	`Remote Browser`	Tile title.
`server_url`	string	(Settings)	Feature Server/Core WebSocket URL (e.g. `ws://192.168.0.114:400` If `session_id` is provided, must include `?token=...`.
`initial_url`	string	`about:blank`	URL to load when session starts.
`session_id`	string	(optional)	If provided, joins this session (requires `server_url` with `?token=...`). If omitted, creates a new session.

Key	Type	Default	Notes
`video_profile`	string	`auto`	Video quality profile: `auto`, `720p30`, `1080p30`, `1080p60`.
`auto_connect`	bool	`true`	Whether to automatically connect when the tile is created.
`show_overlay`	bool	`true`	Show URL bar and stats overlay.
`show_cursor`	bool	`true`	Show cursor position indicator.
`x`, `y`	number	-	Optional position (fractional 0-1 or pixels).
`width`, `height`	number	-	Optional size (fractional 0-1 or pixels).
`dark_mode`	bool	`false`	Enable dark mode for the browser session.
`opacity`	number	`1.0`	Tile opacity.
`response_topic`	string	`kingkiosk/{device}/system/response`	Response destination.

Remote Browser Control Commands:

Command	Description	Required Keys
`connect_remote_browser`	Connect (creates session if needed)	`window_id`
`disconnect_remote_browser`	Disconnect from the session	`window_id`

Command	Description	Required Keys
`configure_remote_browser`	Update configuration	`window_id`, optional: `server_url`, `initial_url`, `session_id`, `video_profile`, `dark_mode`
`navigate_remote_browser`	Navigate to URL	`window_id`, `url` (http/https only)
`remote_browser_back`	Go back in history	`window_id`
`remote_browser_forward`	Go forward in history	`window_id`
`remote_browser_reload`	Reload current page	`window_id`
`remote_browser_click`	Simulate mouse click (at current pointer position; x/y are accepted for compatibility)	`window_id`, optional: `x`, `y`, `button` (`left`/`right`/`middle`)
`remote_browser_scroll`	Scroll the page	`window_id`, `delta_x`, `delta_y`
`remote_browser_key`	Send key press	`window_id`, `key` (DOM code), optional: `modifiers` (accepted for compatibility)
`remote_browser_text`	Input text directly	`window_id`, `text` (max 10,000 chars)
`remote_browser_clear_data`	Clear cookies/localStorage and restart session	`window_id`
`delete_remote_browser`	Remove the tile	`window_id`
`remove_remote_browser`	Alias for `delete_remote_browser`	`window_id`
`list_remote_browsers`	List all remote browser tiles	(none)
`remote_browser_status` / `get_remote_browser_status`	Debug/status snapshot (tracks, consumers, session)	optional: `window_id`

Browser Persistence: - Browser state (cookies, localStorage, IndexedDB) persists automatically across sessions - Each tile has an isolated persistence profile (different tiles don't share cookies) - Use `remote_browser_clear_data` to clear all persisted data (useful for "logout" functionality)

Security Notes: - URL navigation is restricted to `http://` and `https://` schemes only (`javascript:`, `file:`, `data:` blocked) - Text input is limited to 10,000 characters to prevent abuse - Pointer coordinates are clamped to prevent overflow (-100 to 4096)

Example - Create Remote Browser:

```
{
  "command": "create_remote_browser",
  "window_id": "browser_1",
  "name": "Web Browser",
  "server_url": "ws://192.168.0.114:4000/ws",
  "initial_url": "https://www.google.com",
  "video_profile": "720p30",
  "auto_connect": true,
  "show_overlay": true
}
```

Example - Navigate to URL:

```
{
  "command": "navigate_remote_browser",
  "window_id": "browser_1",
  "url": "https://www.example.com"
}
```

Example - Send Key Press:

```
{
  "command": "remote_browser_key",
  "window_id": "browser_1",
  "key": "Enter",
  "modifiers": ["ctrl"]
}
```

Example - Configure with Session ID:

```
{
  "command": "configure_remote_browser",
  "window_id": "browser_1",
  "session_id": "new_session_xyz",
  "video_profile": "1080p30"
}
```

Example - Clear Browser Data (Logout/Reset):

```
{
  "command": "remote_browser_clear_data",
  "window_id": "browser_1"
}
```

This clears all cookies, localStorage, and other persisted browser state, then restarts the session. Useful for implementing "logout" functionality when using web apps that store auth tokens in cookies/localStorage.

Element-Level Commands (via `kingkiosk/{device_id}/element/{window_id}/cmd`):

The remote browser controller also supports element-scoped commands:

Command	Description	Payload Keys
`configure`	Configure the browser	optional: `server_url`, `initial_url`, `session_id`, `video_profile`, `dark_mode`
`connect`	Connect (creates session if needed)	(none)
`disconnect`	Disconnect from session	(none)
`navigate` / `goto`	Navigate to URL	`url` (http/https only)
`back`	Go back	(none)
`forward`	Go forward	(none)
`reload`	Reload page	(none)
`click`	Simulate click at current pointer position	`button` (optional, default `left`)
`scroll`	Scroll page	`dx`, `dy`
`key`	Send key	`code` (DOM KeyboardEvent.code)
`text`	Input text	`text` (max 10,000 chars)
`widget_command`	Forward a command to a widget inside the remote browser (Custom Widget Bridge)	`widget_command` (string), optional: `payload` (object)

`widget_command` also supports nested payload shape: `{"command":"widget_command","payload":{"c`

State Published (on `kingkiosk/{device_id}/element/{window_id}/state`):

```json
{
  "type": "remoteBrowser",
  "widget_id": "browser_1",
  "server_url": "ws://192.168.0.114:4000/ws?token=REDACTED",
  "session_id": "abc123",
  "video_profile": "720p30",
  "dark_mode": false,
  "connected": true,
  "consuming": true,
  "has_control": true,
  "current_url": "https://www.google.com",
  "load_state": "complete",
  "stats": {
    "rtt_ms": 25,
    "fps": 30,
    "bitrate_kbps": 2500,
    "loss_pct": 0.1
  },
  "error": null
}
```

Input Mapping (tvOS/Apple TV Remote):

Input	Action
D-pad	Move pointer (with acceleration)
Select/Enter	Click at current pointer position
Menu/Escape	Navigate back
Play/Pause	Send Space key
Touch swipe	Scroll
Long press	Right-click (context menu)

0.1.13.3 System (Non-Window) Commands

This section documents **system-level commands that are not tied to a specific window type**. These are sent on `kingkiosk/{device_id}/system/cmd`.

Unless explicitly stated, these commands use the payload key `command` to select the handler.

0.1.13.3.1 Volume (`set_volume`, `mute`, `unmute`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	`set_volume`, `mute`, `unmute`.
`value`	number/string	-	Only used by `set_volume`. Parsed as double in range `[0.0, 1.0]`.
`response_topic`	string	`kingkiosk/{device}/system/response`	Response is always published.

Response payloads:

- `set_volume`: `{success, command:'set_volume', volume, timestamp}`
- `mute`/`unmute`: `{success, command:'mute'|'unmute', timestamp}`

0.1.13.3.2 Brightness (`set_brightness`, `get_brightness`, `restore_brightness`, `request_brightness_permission`, `check_brightness_permission`, `resume_kiosk_after_permis`

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	One of the brightness commands above.
`value`	number/string	-	Used by `set_brightness`. Parsed as double in range `[0.0, 1.0]`.
`response_topic`	string	`kingkiosk/{device}/system/response`	Used by most brightness actions.

Notes:

- Brightness is implemented as **application brightness** (not global/system brightness).
- `get_brightness` publishes to `response_topic` **only when `response_topic` is provided** and returns `{brightness, type:'application'}`.
- `request_brightness_permission` / `check_brightness_permission` always return `permission_granted: true`.
- `resume_kiosk_after_permission` performs Android-only behavior; on non-Android it returns `{not_applicable:true}`.

0.1.13.3.3 Notifications (`notify`, `alert`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	`notify` or `alert`.
`title`	string	`MQTT Notification` / `Alert`	-
`message` / `body`	string	(required)	Message body (either key accepted).
`response_topic`	string	`kingkiosk/{device}/system/response`	Handler publishes a result payload.

`notify` additional keys:

Key	Type	Default	Notes
`duration` / `duration_seconds` / `toast_duration`	number	-	Auto-dismiss seconds (Flutter toast + tvOS banner).
`priority`	string	`normal`	One of `low`, `normal`, `high` (platform-dependent).

Key	Type	Default	Notes
format / message_format	string	plain	plain, markdown, segments (tvOS supports markdown + segments; HTML is not required).
markdown / message_markdown / body_markdown	string	-	Convenience: markdown content (if set, treated as format: markdown).
segments / rich_segments	array	[]	When format: "segments": list of { text, bold?, italic?, underline?, color?, font_size? }.
thumbnail / image_url / imageUrl / image	string	-	Optional image URL shown in the banner (tvOS supports all keys).
is_html / html	bool	false	HTML rendering is platform-dependent (tvOS currently uses markdown/segments instead).

alert additional keys:

Key	Type	Default	Notes
`type`	string	`info`	Used to derive default priority if `priority` is not set (`error`/`warning`/`info`/`success`)
`priority`	string	(derived)	If provided, overrides type-derived priority (`low`/`normal`/`high`).
`position`	string	`center`	String forwarded to the alert UI (implementation supports positioned alerts).
`show_border`	bool	`true`	Border shown unless explicitly set to `false`.
`border_color`	string	-	#RRGGBB or #AARRGGBB (optional).
`auto_dismiss_seconds`	int/string	-	Optional auto-dismiss; clamped to `[1, 300]`.
`format` / `message_format`	string	`plain`	`plain`, `markdown`, `segments` (same rich text support as `notify`).
`markdown` / `message_markdown` / `body_markdown`	string	-	Convenience: markdown content (if set, treated as `format: markdown`).

Key	Type	Default	Notes
`segments` / `rich_segments`	array	`[]`	When `format: "segments"`: list of `{ text, bold?, italic?, underline?, color?, font_size? }`.
`is_html` / `html`	bool	`false`	Treat message as HTML.
`thumbnail` / `image_url` / `imageUrl` / `image`	string	-	Network image URL.

0.1.13.3.4 Halo Effect (`halo_effect`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	Must be `halo_effect`.
`window_id`	string	-	If provided, applies halo to a specific window; otherwise applies global halo.
`enabled`	bool	`true`	If `false`, disables the halo (global or window-scoped).
`color`	string/int	`#FF0000`	Hex string (parsed) or ARGB int. Defaults to red.

Key	Type	Default	Notes
`width`	number/string	-	Clamped to `[1.0, 200.0]` if provided.
`intensity`	number/string	-	Clamped to `[0.0, 1.0]` if provided.
`pulse_mode`	string	`none`	One of `none`, `gentle`, `moderate`, `alert`.
`pulse_duration`	int/string	`2000`	Duration (ms), clamped to `[100, 10000]`.
`fade_in_duration`	int/string	`800`	Duration (ms), clamped to `[50, 5000]`.
`fade_out_duration`	int/string	`1000`	Duration (ms), clamped to `[50, 5000]`.
`confirm`	bool	`false`	If true, publishes a confirmation payload (see below).
`response_topic`	string	`kingkiosk/{device}/system/response`	Always publishes `{success, command:'halo_effect', window_id?, timestamp}`.

Confirmation topics (only when `confirm == true`):

- Global: `kingkiosk/{device}/halo_effect/status`
- Window-scoped: `kingkiosk/{device}/window/{window_id}/halo_effect/status`

0.1.13.3.5 Screensaver (`screensaver`) A full-screen overlay with independently bouncing items (clock, image, text, icon). Sits on top of all content when enabled. Tap anywhere to dismiss.

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	Must be `screensaver` or `screen_saver`.
`action`	string	`enable`	One of `enable` (aliases: `on`, `start`), `disable` (aliases: `off`, `stop`), `toggle`, `wake`, `wake_up`, `deactivate`, `set_config` (alias: `configure`), `set_items`, `add_item`, `remove_item`, `update_item`, `clear_items`, `get_state`.
`items`	array	-	Array of screensaver item objects (see below). Used with `enable` or `set_items`.
`item`	object	-	Single screensaver item object. Used with `add_item`.
`item_id`	string	-	Item ID to target. Used with `remove_item` or `update_item`.
`config`	object	-	Config updates for `update_item`.

Key	Type	Default	Notes
`background_color`	string/int	`#000000`	Hex string or ARGB int for background.
`background_opacity`	number	`0.9`	Background opacity (0.0-1.0).
`idle_timeout`	int	`0`	Seconds of inactivity before auto-enable (0 = manual only).

Screensaver Item Object:

Key	Type	Default	Notes
`id`	string	auto-generated	Unique identifier for the item.
`type`	string	`text`	One of `clock`, `image`, `text`, `icon`, `logo`.
`config`	object	`{}`	Type-specific configuration (see below).
`width`	number	`150`	Base width in logical pixels.
`height`	number	`80`	Base height in logical pixels.
`speed`	number	`1.0`	Movement speed multiplier (0.1-3.0). Higher = faster bouncing.
`scale`	number	`1.0`	Size scale factor (0.5-3.0).

Type-specific config:

- **clock**: `{ "show_seconds": true, "show_date": false, "font_size": 48, "text_color": "#FFFFFF" }`
- **image/logo**: `{ "url": "https://example.com/logo.png", "fit": "contain" }`
- **text**: `{ "text": "Hello", "font_size": 36, "font_weight": "bold", "text_color": "#FFFFFF" }`
- **icon**: `{ "icon": "star", "size": 64, "color": "#FFFFFF" }` (icons: star, heart, home, settings, music, play, pause, stop, cloud, sun, moon)

Example: Enable screensaver with bouncing clock and logo

```json
{
  "command": "screensaver",
  "action": "enable",
  "items": [
    {
      "id": "clock_1",
      "type": "clock",
      "config": { "show_seconds": true, "text_color": "#00FF00" },
      "width": 250,
      "height": 100,
      "speed": 1.0,
      "scale": 1.5
    },
    {
      "id": "logo_1",
      "type": "image",
      "config": { "url": "https://example.com/logo.png" },
      "width": 200,
      "height": 200,
      "speed": 0.7,
      "scale": 1.0
    }
  ],
  "background_color": "#000000",
  "background_opacity": 0.95
}
```

Example: Disable screensaver

```json
{
  "command": "screensaver",
  "action": "disable"
}
```

Example: Add a text item to running screensaver

```
{
  "command": "screensaver",
  "action": "add_item",
  "item": {
    "id": "welcome_text",
    "type": "text",
    "config": { "text": "Welcome!", "font_size": 48, "text_color": "#FF6600" },
    "width": 300,
    "height": 80,
    "speed": 1.2
  }
}
```

Example: Get current screensaver state

```
{
  "command": "screensaver",
  "action": "get_state"
}
```

Response includes full state: `{ "enabled": true, "items": [...], "background_color": "#000000", ... }`

Idle Screensaver (Settings-Based):

King Kiosk also includes an idle-based screensaver that activates automatically after a configurable timeout period. This is configured in **Settings → App Settings → Screensaver** with three modes:

Mode	Behavior
`off`	Idle screensaver disabled
`dim`	Screen goes black after timeout
`screensaver`	Bouncing clock appears after timeout

The idle screensaver timeout is configurable from 1-60 minutes.

Example: Wake from idle screensaver

Use this command to remotely dismiss the idle screensaver (whether in dim or clock mode):

```
{
  "command": "screensaver",
  "action": "wake"
}
```

Alternative actions: `wake_up`, `deactivate`

This command: 1. Disables any active MQTT-triggered bouncing screensaver 2. Deactivates the idle-based screensaver (restores brightness for dim mode, hides bouncing clock for screensaver mode) 3. Resets the idle timer so the screensaver won't immediately reactivate

---

0.1.13.3.6 Settings / FAB Lock (`lock_fab`, `unlock_fab`, `lock_settings`, `unlock_settings`)

PIN-protected remote lock/unlock for the settings FAB.

`lock_fab` and `lock_settings` are equivalent aliases.
`unlock_fab` and `unlock_settings` are equivalent aliases.

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	One of `lock_fab`, `unlock_fab`, `lock_settings`, `unlock_settings`.
`pin` / `settings_pin` / `settingsPin` / `code`	string/int	(required)	Settings PIN to authorize the action.
`response_topic`	string	`kingkiosk/{device}/system/response`	Response published via unified response helper.
`correlation_id`	string	-	Optional request correlation ID; echoed in responses.

Behavior:

- Both lock and unlock commands require a valid settings PIN.
- Lock commands set settings to locked and drive the normal FAB melt/ember transition.
- Unlock commands set settings to unlocked and drive the normal reveal/awake transition.

- Transitions continue from the current visual state (no forced reset), including current ember/menu workflow.
- If no custom settings PIN is configured on device, the runtime fallback PIN is `1234`.

Response payloads:

- Success: `{success:true, status:'success', command, message, locked, timestamp, device, ...}`
- Error: `{success:false, status:'error', command, error, timestamp, device, ...}`
- Common errors: unsupported command, missing PIN, invalid PIN, settings controller unavailable.

Example: lock FAB

```
{
  "command": "lock_fab",
  "pin": "1234"
}
```

Example: unlock FAB (alias + alternate PIN key)

```
{
  "command": "unlock_settings",
  "settings_pin": "1234"
}
```

0.1.13.3.7 Person Detection (`person_detection`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	Must be `person_detection`.
`action`	string	`toggle`	One of `enable`, `disable`, `toggle`, `status`.

Key	Type	Default	Notes
`confirm`	bool	`false`	If true, publishes a confirmation payload.

Published topics:

- Always publishes current status to `kingkiosk/{device}/person_presence`.
- If `confirm == true`, also publishes to `kingkiosk/{device}/person_detection/status`.

0.1.13.3.8 Security Camera (`security_camera`)

Note: **Local settings are authoritative.** If the Security Camera is disabled in the app's Settings, MQTT requests to enable it or change its interval will be rejected.

Controls the periodic "security camera" capture flow in the WebRTC media service.

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	Must be `security_camera`.
`action`	string	-	One of `enable`, `disable`, `set_interval`, `status`.
`interval`	int/string	3	Used by `enable` and `set_interval` (seconds).

Published topics:

- When enabled, publishes security camera snapshots to:
  - `kingkiosk/{device}/camera/snapshot` (raw PNG bytes, retained)

    - `kingkiosk/{device}/camera/state` (JSON metadata, retained)

- For `action == status`, additionally publishes to `kingkiosk/{device}/security_camera/statu` with `{enabled, interval_seconds}`.

Responses:

- Also publishes a standardized response to `kingkiosk/{device}/system/response` via the unified response helper.

0.1.13.3.9 Screenshot Camera (`screenshot_camera`)

Controls the periodic "screenshot camera" capture flow in the screenshot service.

Note: **Local settings are authoritative.** If Screenshot Camera is disabled in the app's Settings, MQTT requests to enable it or change its interval will be rejected.

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	Must be `screenshot_camera`.
`action`	string	-	One of `enable`, `disable`, `set_interval`, `status`.
`interval`	int/string	5	Used by `enable` and `set_interval` (seconds).

Published topics:

- When enabled, publishes screenshot camera snapshots to:
    - `kingkiosk/{device}/screenshot/snapshot` (raw PNG bytes, not retained)
    - `kingkiosk/{device}/screenshot/state` (JSON metadata, not retained)

- For `action == status`, additionally publishes to `kingkiosk/{device}/screenshot_camera/sta` with `{enabled, interval_seconds}`.

Responses:

- Also publishes a standardized response to `kingkiosk/{device}/system/response` via the unified response helper.

0.1.13.3.10 Screenshot (`screenshot`)

Note: **Local settings are authoritative.** If screenshots are disabled locally (Screenshot Camera is OFF in Settings), MQTT screenshot requests will be rejected.

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	Must be `screenshot`.
`notify`	bool	`false`	If true, shows an on-device UI snackbar.
`confirm`	bool	`false`	If true, publishes to `kingkiosk/{device}/screens` on success or error (independent of Home Assistant discovery).

Published topics:

- When Home Assistant discovery is enabled, publishes screenshot payload to `kingkiosk/{device}/scree` as a **raw base64 string** (base64-encoded PNG bytes, retained).
- If `confirm == true`, publishes a status JSON payload to `kingkiosk/{device}/screenshot/status`

Decoding tip:

- Avoid `mosquitto_sub -v` (it prefixes the topic, breaking base64 decode).

- Capture exactly one payload and decode:
  - `mosquitto_sub -t 'kingkiosk/<device>/screenshot' -C 1 -R > shot.b64`
  - macOS: `base64 -D shot.b64 > shot.png`
  - Linux: `base64 -d shot.b64 > shot.png`

0.1.13.3.11 Cache (`cache`, `cache_control`, `clear_cache`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	One of `cache`, `cache_control`, `clear_cache` (all route here).
`action`	string	`stats`	See action list below.
`url`	string	-	Required for `refresh` / `refresh_resource`.
`response_topic`	string	`kingkiosk/{device}/system/response`	Response published via the unified response helper.

Supported `action` values:

- `clear`, `clear_all`, `nuclear`
- `clear_images`
- `clear_data`
- `refresh`, `refresh_resource` (requires `url`)
- `stats`, `get_stats`

0.1.13.3.12 Text-to-Speech (`tts`, `speak`, `say`)

These commands forward an action map into the TTS service.

Feature Server transparent takeover: When the Feature Server is connected, `speak`, `getVoices`, and `status` commands automatically route through the Feature Server's high-quality Piper TTS engine. When disconnected, the same commands fall back to on-device TTS (FlutterTts on Flutter, AVSpeechSynthesizer on tvOS). No changes to the MQTT command format are required — the routing is transparent.

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	`tts`, `speak`, or `say`.
`response_topic`	string	-	If provided, publishes the TTS service result.
`queue_first`	bool	`true`	Defaulted to `true` by the system handler (unless explicitly `false`). Also controls pre-init queuing behavior in the service.

TTS action selection:

- The TTS service uses `action` (preferred), or falls back to `command`.

Common TTS keys:

Key	Type	Default	Notes
`action`	string	`speak`	Supported actions listed below.
`text` / `message`	string	-	Used by `speak`/`say`/`tts` actions.
`language`	string	-	Example: `en-US`.

Key	Type	Default	Notes
`voice`	string	-	Voice name. When Feature Server is connected, use a Piper voice ID (e.g. `en_US-lessac-medium`).
`volume`	number	-	0.0–1.0. Mapped to 0–100 for Feature Server.
`speechRate` / `rate`	number	-	0.0–1.0. Mapped to 0–100 for Feature Server.
`pitch`	number	-	0.5–2.0. Mapped to 0–100 for Feature Server (1.0 = 33).
`queue`	bool	`false`	If true (or if already speaking), queues the speak.
`force`	bool	`false`	If true, bypasses deduplication check.
`dedupe_ms` / `dedupeMs` / `dedupe_window_ms`	int	`1200`	Deduplication window in milliseconds. If the same text+language+voice fingerprint is sent within this window, the duplicate is silently skipped.

Feature Server-only speak keys (ignored when using on-device TTS):

Key	Type	Default	Notes
`delivery_mode`	string	`url`	`url` (recommended) or `inline` (base64 in WS notification).
`speaker_id`	string	-	Multi-speaker voice speaker selection.
`appended_silence_ms`	int	-	Silence appended after synthesis (ms).

Supported TTS `action` values:

- Speak: `tts`, `speak`, `say`
- Playback control: `stop`, `pause`, `resume`
- Settings: `setVolume/volume`, `setRate/rate/speed`, `setPitch/pitch`, `setLanguage/language`, `setVoice/voice` (each accepts the specific param or generic `value` key)
- Service toggles: `enable`, `disable`
- Info: `status/getStatus`, `getLanguages`, `getVoices/voice_list/voices`
- Queue: `clearQueue`
- Feature Server only: `voice_pull/pull_voice/install_voice`

Feature Server voice actions:

`getVoices` / `voice_list` — When Feature Server is connected, queries available Piper voices. Optional filter keys:

Key	Type	Notes
`language` / `language_code`	string	Filter by language (e.g. `en_US`).
`quality`	string	Filter by quality (`x_low`, `low`, `medium`, `high`).
`query`	string	Free-text search filter.
`installed_only`	bool	Only return installed voices.
`limit`	int	Max voices to return (default 25).

Response includes `voices` array with objects containing `voiceId`, `name`, `languageCode`, `quality`, `installed`, `numSpeakers`, etc.

`voice_pull` — Ensures a voice is downloaded/installed on the Feature Server. Requires `voice` parameter (e.g. `en_US-lessac-medium`). Only available when Feature Server is connected.

`status` / `getStatus` — When Feature Server is connected, response includes `source: "feature_server"`, `engine: "piper"`, and `feature_server_connected: true`.

0.1.13.3.13 Speech-to-Text (`stt`, `speech_to_text`, `listen`) These commands forward an action map into the Speech-to-Text service.

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	`stt`, `speech_to_text`, or `listen`.
`response_topic`	string	-	If provided, publishes the STT service result.
`action`	string	`start`	Action forwarded to the STT service.

Supported STT `action` values and keys:

Action	Keys	Notes
`start` / `listen`	-	Starts listening.
`stop`	-	Stops listening and returns `{text, confidence}`.
`status`	-	Returns service status.
`enable` / `disable`	-	Enables/disables service.
`set_language`	`language`	Sets language (e.g., en).

Action	Keys	Notes
`use_whisper`	`use_whisper` (bool)	IO only; web always uses Web Speech.
`set_mqtt_publishing` / `publish_to_mqtt`	`enabled` (bool)	Controls transcription MQTT publishing.
`set_send_to_ai_agent` / `send_to_ai_agent` / `ai_integration`	`enabled` (bool)	Controls AI agent integration.
`provision_ai_chatbot_only`	-	Sets `send_to_ai_agent=true` and `publish_to_mqtt=false`.

0.1.13.3.14 Audio Input Device (`unified_audio`, `audio_input`, `audio_devices`)

These commands query and control the **audio input device** used by Speech-to-Text (and other UnifiedAudioService consumers).

This is the same device you select in the UI under **Settings → Speech & AI → Audio Input Device**.

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	`unified_audio`, `audio_input`, or `audio_devices`.
`action` / `command_action`	string	`status`	One of `status` (alias: `getstatus`), `list_devices` (aliases: `list`, `devices`), `set_device` (alias: `setdevice`).

Key	Type	Default	Notes
device_id / deviceId	string	-	Required for set_device.
response_topic	string	-	If provided, publishes the action result payload.

Examples:

```
{ "command": "unified_audio", "action": "status", "response_topic":
↪  "kingkiosk/<device>/system/response" }
```

```
{ "command": "unified_audio", "action": "list_devices", "response_topic":
↪  "kingkiosk/<device>/system/response" }
```

```
{ "command": "unified_audio", "action": "set_device", "device_id": "1", "response_topic":
↪  "kingkiosk/<device>/system/response" }
```

0.1.13.3.15 Background (set_background, get_background)

set_background keys:

Key	Type	Default	Notes
command	string	(required)	Must be set_background.
type	string	-	One of default, image, webview.
image_path / image_url	string	-	Used when type == image.
web_url / url	string	-	Used when type == webview.

Key	Type	Default	Notes
`response_topic`	string	-	If provided, publishes `{success, message, type, image_path, web_url}`.

`get_background` keys:

Key	Type	Default	Notes
`command`	string	(required)	Must be `get_background`.
`response_topic`	string	`kingkiosk/{device}/status/background`	Response payload: `{ success: bool, background: { type, image_path, web_url } }`.

0.1.13.3.16 Provision (`provision`) Provision applies settings and can optionally import saved layouts so you can clone one device to another in a single command.

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	Must be `provision`.
`settings`	object	-	Optional. If present, settings are read from this object.

Key	Type	Default	Notes
`screen_states` / `screenStates`	array	-	Optional list of screen-state objects to import.
`screen_state`	object	-	Optional single screen-state object to import.
`current_layout` / `currentLayout`	object	-	Optional current layout snapshot to apply immediately.
`overwrite`	bool	`true`	Used when importing screen states via provision.
`correlation_id`	string	-	Optional. Echoed in the provision response.
`response_topic`	string	`kingkiosk/{device}/system/response`	Provision always publishes a response.
(any other keys)	any	-	If `settings` is not provided, non-reserved keys are treated as settings. Reserved keys include command/response/correlation/import flags and layout payload keys above.

Supported settings keys (case-insensitive, with common aliases):

Setting key(s)	Type	Notes
`isDarkMode`, `darkMode`, `dark_mode`	bool	Theme.

Setting key(s)	Type	Notes
`kioskMode`, `kiosk_mode`	bool	Kiosk mode toggle.
`showSystemInfo`, `show_system_info`	bool	Toggle system info overlay.
`kioskStartUrl`, `kiosk_start_url`, `startUrl`	string	Start URL.
`mqttEnabled`, `mqtt_enabled`	bool	MQTT enable.
`mqttBrokerUrl`, `mqtt_broker_url`, `brokerUrl`	string	Broker host/url.
`mqttBrokerPort`, `mqtt_broker_port`, `brokerPort`	int	1–65535.
`mqttUsername`, `mqtt_username`	string	Stored in secure storage.
`mqttPassword`, `mqtt_password`	string	Stored in secure storage.
`deviceName`, `device_name`	string	Sanitized and applied to MQTT device namespace.
`mqttHaDiscovery`, `mqtt_ha_discovery`, `haDiscovery`	bool	Also updates runtime discovery flag.
`mqttUseSSL`, `mqtt_use_ssl`, `mqttSSL`, `mqtt_ssl`	bool	Enabling may auto-adjust the port.
`mqttAllowSelfSigned`, `mqtt_allow_self_signed`, `mqttSelfSigned`, `mqtt_self_signed`	bool	-
`mqttUseHmacAuth`, `mqtt_use_hmac_auth`, `mqttHmacAuth`, `mqtt_hmac_auth`	bool	Enables/disables MQTT HMAC auth mode.
`mqttHmacSecret`, `mqtt_hmac_secret`, `hmacSecret`	string	Shared secret used by HMAC auth.

Setting key(s)	Type	Notes
`webviewAllowInvalidCerts`, `webview_allow_invalid_certs`, `allowInvalidWebviewCerts`	bool	WebView hardening flag.
`networkAllowInvalidCerts`, `network_allow_invalid_certs`, `allowInvalidCerts`	bool	Network hardening flag for non-WebView requests.
`enableEvalJs`, `enable_evaljs`, `evalJsEnabled`, `evaljs_enabled`	bool	WebView hardening flag.
`mqttCaCertPath`, `mqtt_ca_cert_path`, `mqttCaCert`, `mqtt_ca_cert`	string	-
`mqttClientCertPath`, `mqtt_client_cert_path`, `mqttClientCert`, `mqtt_client_cert`	string	-
`mqttClientKeyPath`, `mqtt_client_key_path`, `mqttClientKey`, `mqtt_client_key`	string	-
`personDetectionEnabled`, `person_detection_enabled`, `personDetection`, `person_detection`	bool	Also updates person detection service state when available.
`haAccessToken`, `ha_access_token`, `homeAssistantToken`, `home_assistant_token`	string	Stored in secure storage and synced to AI agent service when available.
`settingsPin`, `settings_pin`, `pin`	string	Minimum length 4; stored in secure storage.
`sendToAIAgent`, `send_to_ai_agent`, `aiIntegration`, `ai_integration`	bool	Enables/disables Speech-to-AI integration.

Setting key(s)	Type	Notes
`aiAgentEnabled`, `ai_agent_enabled`	bool	Enables/disables AI agent service; enabling may also enable Speech-to-AI.
`aiEnabled`, `ai_enabled`	bool	AI feature toggle.
`aiProviderHost`, `ai_provider_host`, `aiProviderUrl`	string	AI provider endpoint.
`haBaseUrl`, `ha_base_url`, `homeAssistantUrl`, `home_assistant_url`	string	Home Assistant base URL.
`haAgentId`, `ha_agent_id`, `conversationAgent`, `conversation_agent`	string	Home Assistant conversation agent id.
`sipEnabled`, `sip_enabled`	bool	SIP enable toggle.
`sipServerHost`, `sip_server_host`	string	SIP server host.
`sipProtocol`, `sip_protocol`	string	`ws` or `wss`.
`selectedAudioInput`, `selected_audio_input`	string	Selected audio input id/name.
`selectedVideoInput`, `selected_video_input`	string	Selected video input id/name.
`selectedAudioOutput`, `selected_audio_output`	string	Selected audio output id/name.
`wyomingHost`, `wyoming_host`	string	Wyoming host.
`wyomingPort`, `wyoming_port`	int	Wyoming port.
`wyomingEnabled`, `wyoming_enabled`	bool	Wyoming enable toggle.
`featureServerEnabled`, `feature_server_enabled`	bool	Enable/disable Feature Server.
`featureServerAutoConnect`, `feature_server_auto_connect`	bool	Auto-connect Feature Server when app starts.

Setting key(s)	Type	Notes
`featureServerUrl`, `feature_server_url`	string	Feature Server host/IP (cross-platform safe format; avoid `ws://` prefix).
`featureServerUseHttps`, `feature_server_use_https`	bool	Use secure WebSocket (`wss`) for Feature Server signaling.
`featureServerProduceAudio`, `feature_server_produce_audio`	bool	Include microphone audio when producing to Feature Server.
`intercomEnabled`, `intercom_enabled`	bool	Enable intercom/broadcast participation.
`intercomGroups`, `intercom_groups`	array	Intercom groups (tvOS supports this directly; other clients may ignore).
`websocketUrl`, `websocket_url`	string	Websocket endpoint.
`mediaServerUrl`, `media_server_url`	string	Media server endpoint.
`latestScreenshot`, `latest_screenshot`	string	Metadata/path field.
`autoLockEnabled`, `auto_lock_enabled`	bool	Enable/disable auto-lock for settings screen.
`autoLockTimeout`, `auto_lock_timeout`, `autoLockTimeoutMinutes`, `auto_lock_timeout_minutes`	double	Auto-lock timeout in minutes (e.g. 1, 2, 5, 10, 15, 30, 60).

Setting key(s)	Type	Notes
`screensaverMode`, `screensaver_mode`	string	Screensaver mode: `off`, `dim`, or `clock` (Flutter) / `screensaver` (tvOS).
`screensaverTimeout`, `screensaver_timeout`, `screensaverTimeoutMinutes`, `screensaver_timeout_minutes`	double	Screensaver timeout in minutes.
`backgroundType`, `background_type`	string	Background type: `default`, `image`, or `webview`.
`backgroundImageUrl`, `background_image_url`, `backgroundImagePath`, `background_image_path`	string	Background image URL/path.
`backgroundWebUrl`, `background_web_url`	string	Background WebView URL (Flutter only).
`locationEnabled`, `location_enabled`	bool	Enable location services (Flutter only).
`brightnessLevel`, `brightness_level`	double	Screen brightness 0–100 (Flutter only).
`mqttReconnectOnStartup`, `mqtt_reconnect_on_startup`	bool	Auto-reconnect MQTT on app startup.
`kingDspDiscoveryEnabled`, `kingdsp_discovery_enabled`, `kingDspIntercomEnabled`	bool	Enable KingDSP network discovery.
`dlnaRendererEnabled`, `dlna_renderer_enabled`	bool	Enable DLNA/UPnP media renderer.

Setting key(s)	Type	Notes
`enableContinuityCamera`	bool	Enable Continuity Camera (tvOS only).
`sttEnabled`, `stt_enabled`, `speechToTextEnabled`, `speech_to_text_enabled`	bool	Enable speech-to-text service (Flutter).
`autoSpeakResponses`, `auto_speak_responses`	bool	Auto-speak AI responses (Flutter).
`continueListening`, `continue_listening`	bool	Continue listening after AI response (Flutter).
`keepConversationHistory`, `keep_conversation_history`	bool	Keep AI conversation history (Flutter).
`ttsRate`	float	TTS speech rate (tvOS, 0.0–1.0).
`ttsPitch`	float	TTS speech pitch (tvOS, 0.5–2.0).
`ttsVolume`	float	TTS speech volume (tvOS, 0.0–1.0).
`ttsLanguage`	string	TTS language code e.g. `en-US` (tvOS).
`enablePersonDetection`	bool	Enable person detection (tvOS).
`enableFaceRecognition`	bool	Enable face recognition (tvOS).
`detectionInterval`	double	ML detection interval in seconds (tvOS).
`enableKingDSP`	bool	Enable KingDSP audio streaming (tvOS).

Setting key(s)	Type	Notes
`kingDSPHost`	string	KingDSP server host (tvOS).
`kingDSPPort`	int	KingDSP server port (tvOS, default 4954).
`aiProvider`	string	AI provider: `openai`, `anthropic`, `google`, `custom`, `homeassistant` (tvOS).
`aiModel`	string	AI model name (tvOS).
`aiApiKey`	string	AI API key; stored in secure storage (tvOS).
`aiSystemPrompt`	string	AI system prompt (tvOS).
`aiMaxTokens`	int	AI max tokens (tvOS).
`aiTemperature`	double	AI temperature 0.0–2.0 (tvOS).
`aiAutoSpeak`	bool	Auto-speak AI responses (tvOS).
`aiListenAfterResponse`	bool	Continue listening after AI response (tvOS).
`aiKeepHistory`	bool	Keep conversation history (tvOS).
`sttLanguage`	string	STT language code (tvOS).
`sttModel`	string	STT model name (tvOS).

Setting key(s)	Type	Notes
`sttTranslate`	bool	Translate STT to English (tvOS).
`sttUseVAD`	bool	Use Voice Activity Detection for STT (tvOS).
`ai_chatbot` / `ai_chat_bot` / `chatbot`	object	See AI Chat Bot object below.

Provision response payload includes:

- `status`: `success`, `partial`, or `error`
- `applied_settings`: list of settings applied
- `failed_settings`: map of setting key -> reason
- `screen_states_imported`: imported state names
- `screen_states_failed`: map of state name -> reason
- `current_layout_applied`: bool
- `correlation_id` (when provided in request)

AI Chat Bot object (`ai_chatbot`) keys:

Key	Type	Notes
`provider`	string	-
`api_key`	string	-
`base_url`	string	-
`model`	string	-
`system_prompt`	string	Stored under the `systemPrompt` key in the service config.

Feature Server provisioning example:

```
{
  "command": "provision",
  "settings": {
    "featureServerEnabled": true,
    "featureServerAutoConnect": true,
    "featureServerUrl": "192.168.1.50",
    "featureServerUseHttps": false,
    "featureServerProduceAudio": true,
    "intercomEnabled": true
  },
  "correlation_id": "provision-feature-server-001"
}
```

After provisioning, subscribe to:

`kingkiosk/{device_id}/feature_server/state`

The payload is retained and includes fields like `enabled`, `connected`, `state`, `last_error`, and reconnect metadata.

0.1.13.3.17 Get Config (`get_config`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	Must be `get_config`.
`include_secrets` / `includeSecrets`	bool	`true`	Include secret values (passwords/tokens/PIN/secrets) in returned config.
`include_layouts` / `includeLayouts` / `include_screen_states`	bool	`true`	Include saved screen states and current layout snapshot.
`correlation_id`	string	-	Optional. Echoed in response.
`response_topic`	string	`kingkiosk/{device}/system/response`	

Response payload:

```
{command:'get_config', status:'success', device_name, config:{...},
settings:{...}, timestamp, correlation_id?, screen_states?, screen_state_count?,
current_layout?}
```

Notes:

- `settings` is an alias of `config` for compatibility.
- When `include_secrets` is `false`, secret fields are masked as `***`.
- When `include_layouts` is `true`, response includes `screen_states`, `screen_state_count`, and `current_layout`.
- The `config` object includes all provisionable settings listed in the provision table above (including screensaver, background, auto-lock, networked audio, ML detection, TTS, STT, AI behavior, and MQTT reconnect settings). This allows admin tools to prepopulate settings screens with current device state.

0.1.13.3.18 AI Agent (`ai_agent` / `ai`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	`ai_agent` or `ai`.
`action`	string	-	Required for most operations.

Supported `action` values and keys:

Action	Keys	Published topics / notes
`enable`	`enabled` (bool, default true)	Enables/disables AI agent service.
`select_agent`	`agent_index` (int)	Selects an agent by index.

Action	Keys	Published topics / notes
configure_home_assistant	base_url, access_token, agent_id (optional)	Auto-discovers agents after config.
send_message	message (string), conversation_id (optional)	Sends text to AI agent.
create_conversation	user_id (optional), user_name (optional)	Publishes kingkiosk/{device}/ai_agent/conversation_created.
switch_user	user_id, user_name	Switches active conversation.
clear_conversation	conversation_id (optional)	Clears one or all conversations.
get_status	-	Publishes kingkiosk/{device}/ai_agent/status.
speech_integration	enabled (bool, default true)	Enables Speech-to-AI integration.
speech_mqtt_publishing / publish_speech_to_mqtt	enabled (bool, default true)	Enables transcription MQTT publishing.
discover_agents	-	Publishes kingkiosk/{device}/ai_agent/agents_discovered.
select_ha_agent	agent_id	Selects HA conversation agent.
configure_chat_bot	provider, api_key, base_url, model	Configures the local Chat Bot.

Additional published topics (implementation detail, but useful for admin UIs):

- `kingkiosk/{device}/ai_agent/message_response` (non-retained): emitted for `send_message` with `message`, `response`, `status`, `timestamp`.
- `kingkiosk/{device}/ai_agent/conversation_cleared` (non-retained): emitted for `clear_conversation`.

0.1.13.3.19 AI Provisioning (`provision_ai_chatbot`, `setup_ai_chatbot`, `configure_ai_chatbot`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	One of the provisioning command aliases above.
`provider`	string	`anthropic`	Supported: `anthropic`, `openai`, `gemini`, `ollama`.
`api_key`	string	-	Provider API key (if needed).
`base_url`	string	-	Used by providers like `ollama`.
`model`	string	(provider default)	Defaults depend on provider.
`system_prompt`	string	(provider default)	Defaults depend on provider.
`enable_speech`	bool	`true`	Enables Speech-to-Text.
`enable_tts`	bool	`true`	Enables Text-to-Speech.

Key	Type	Default	Notes
`chatbot_only_mode`	bool	`true`	Disables MQTT publishing of transcriptions when true.

Publishes status to `kingkiosk/{device}/ai_provisioning/status`.

0.1.13.3.20 Command History / Audit (`mqtt_cmd_history` + aliases)

This subsystem exposes the in-memory command audit/history service over MQTT.

Primary command:

- `mqtt_cmd_history`

Aliases (mapped to `mqtt_cmd_history` internally):

- `get_command_history` (sets `action: 'list'` and defaults `limit` to 100)
- `get_audit_history` (sets `action: 'list'` and defaults `limit` to 100)
- `clear_command_history` (sets `action: 'clear'`)
- `clear_audit_history` (sets `action: 'clear'`)
- `get_audit_stats` (sets `action: 'stats'`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	One of the command strings above.
`action`	string	`list`	Only used when `command == mqtt_cmd_history`. See supported actions below.

Key	Type	Default	Notes
`limit`	int/string	`100`	Used for list and many queries.
`response_topic`	string	`kingkiosk/{device}/system/response`	Responses are published to this topic (non-retained).

Supported `action` values (when using `command: 'mqtt_cmd_history'`):

- `list` / `get`
- `stats`
- `clear`
- `query_by_time` / `query_time_range` (requires `start_time` and `end_time` as ISO8601)
- `query_by_correlation` / `query_correlation` (requires `correlation_id`)
- `query_by_command` / `query_command_type` (requires `command_type`)
- `query_by_source` / `query_source_type` (requires `source_type`)
- `query_by_window` / `query_window` (requires `window_id`)
- `query_by_batch` / `query_batch` (requires `batch_id`)
- `query_by_status` / `query_response_status` (requires `status`)
- `replay` (see below)

Action-specific keys:

Action	Keys
`query_by_time` / `query_time_range`	`start_time` (ISO8601), `end_time` (ISO8601), optional `limit`
`query_by_correlation` / `query_correlation`	`correlation_id`
`query_by_command` / `query_command_type`	`command_type`, optional `limit`
`query_by_source` / `query_source_type`	`source_type` (example values: `mqtt`, `touch`, `batch`, `api`, `local`), optional `limit`
`query_by_window` / `query_window`	`window_id`, optional `limit`
`query_by_batch` / `query_batch`	`batch_id`

Action	Keys
`query_by_status` / `query_response_status`	`status` (expected: `success`, `error`, `pending`), optional `limit`
`replay`	`command_ids` (list of ints/strings) OR `correlation_id`, optional `dry_run` (bool)

Response payloads:

- Responses are published to `response_topic` as JSON with `command: 'audit_response'` and include `action`, `timestamp` (ISO8601), `device`, and action-specific fields.
- Errors are also published to `response_topic` with `action: 'error'` and an `error` string.

0.1.13.3.21 Debug / Introspection (`test_sensors`, `debug_sensors`, `test_location`, `debug_location`, `list_windows`, `debug_windows`)

These are dispatcher-level debug helpers.

0.1.13.3.21.1 Sensors (`test_sensors`, `debug_sensors`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	`test_sensors` or `debug_sensors`.

Behavior:

- Attempts to publish current sensor values via the normal sensor publisher.
- Responds on `kingkiosk/{device}/system/response` using the unified response helper; success includes `sensors_available: true|false`.

0.1.13.3.21.2 Location (`test_location`, `debug_location`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	`test_location` or `debug_location`.

Behavior:

- If sensors are unavailable (WASM mode), publishes an error response.
- Otherwise requests location permission and publishes direct sensor topics:
  - `kingkiosk/{device}/latitude` (retained)
  - `kingkiosk/{device}/longitude` (retained)
  - `kingkiosk/{device}/location_status` (retained)

0.1.13.3.21.3 Window list (`list_windows`, `debug_windows`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	`list_windows` or `debug_windows`.

Response:

- Publishes a success payload to `kingkiosk/{device}/system/response` via the unified response helper with:
  - `window_count`
  - `windows` (visual tile list when available)
  - `tiling_mode` (when available)
  - `controller_count` and `controllers` (debug listing)

Live updates:

- For always-on admin dashboards, prefer the retained window state feed:
  - Snapshot: `kingkiosk/{device}/windows` (retained)
  - Events: `kingkiosk/{device}/windows/event` (non-retained)
  - Diagnostics: `kingkiosk/{device}/diagnostics/windows` (retained)

Each `windows[]` item includes:

- `window_id`, `title`, `type`, `url`, `image_urls`
- `x`, `y`, `width`, `height`, `opacity`, `z_index`
- `loop`, `minimized`, `maximized`
- `mqtt_topic`, `mqtt_json_field`, `mqtt_is_base64`, `mqtt_update_interval_ms`
- `metadata`

0.1.13.3.22 Batch / Script (`batch`, `kill_batch_script`, `batch_status`, `wait`)

The batch subsystem executes a sequence of commands.

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	`batch`, `kill_batch_script`, `batch_status`, or `wait`.
`response_topic`	string	`kingkiosk/{device}/system/response`	Optional override; must remain within `kingkiosk/{device}/...` and must not contain `#`, `+`, or NUL.

`batch` keys:

Key	Type	Notes
`commands`	array	Required. Each item may be a JSON object (with its own `command`) or a string.

`wait` step (in a batch) keys:

Key	Type	Default	Notes
`seconds`	number/string	`1`	Clamped to 0–300 seconds internally (milliseconds clamped to 0–300000).

Standalone `wait` command keys:

Key	Type	Default	Notes
`seconds`	number/string	`1`	Must be `> 0` and `<= 300`.

Notes:

- `batch` does not currently publish an automatic completion event; use per-command responses inside the batch (where supported) or query `batch_status`.
- `batch_status` publishes `{batch_running, batch_id, status, progress, total, kill_requested, ...}`.

0.1.13.3.23 Screen State (`save_screen_state`, `load_screen_state`, `list_screen_states`, `delete_screen_state`, `export_screen_state`, `import_screen_state`)

These commands manage **named saved layouts** (window tiles + layout settings).

All commands in this section support:

- `response_topic` (optional): publish response to a custom topic (defaults to `kingkiosk/{device}/sys`
- `correlation_id` (optional): echoed in the response when provided

Response shape:

`{status, message, timestamp, state_name?, command?, correlation_id?, ...data}`

0.1.13.3.23.1 `save_screen_state`

Key	Type	Default	Notes
`name`	string	(required)	Screen state name.
`overwrite`	bool	`false`	If false and the name exists, returns an error response.
`response_topic`	string	-	Optional response topic.
`correlation_id`	string	-	Optional request/response correlation id.

Success response includes: `name`, `windowCount`, `savedAt`.

0.1.13.3.23.2 `load_screen_state`

Key	Type	Notes
`name`	string	(required)
`response_topic`	string	Optional response topic.
`correlation_id`	string	Optional request/response correlation id.

Success response includes: `name`, `windowCount`, `savedAt`.

0.1.13.3.23.3 `list_screen_states` Optional keys: `response_topic`, `correlation_id`.

Success response includes:

- `states`: list of `{name, windowCount, savedAt}`
- `count`

0.1.13.3.23.4 `delete_screen_state`

Key	Type	Notes
`name`	string	(required)
`response_topic`	string	Optional response topic.
`correlation_id`	string	Optional request/response correlation id.

0.1.13.3.23.5 `export_screen_state`

Key	Type	Notes
`name`	string	(required)
`response_topic`	string	Optional response topic.
`correlation_id`	string	Optional request/response correlation id.

Success response includes: `name`, `windowCount`, `savedAt`, `exportedAt`, and `screen_state` (full exported object).

0.1.13.3.23.6 `import_screen_state`

Key	Type	Default	Notes
`name`	string	(required)	Name to save the imported screen state as (overrides any name inside the imported object).
`screen_state`	object	(required)	The exported screen state object (as produced by `export_screen_state`).

Key	Type	Default	Notes
`overwrite`	bool	`false`	If false and the name exists, returns an error response.
`response_topic`	string	-	Optional response topic.
`correlation_id`	string	-	Optional request/response correlation id.

0.1.13.3.24 Fleet Layout Replication (`replicate_layout`, `subscribe_fleet`, `unsubscribe_fleet`)

These commands publish/receive layout updates on a shared fleet topic.

0.1.13.3.24.1 `replicate_layout`

Top-level keys:

Key	Type	Default	Notes
`fleet_id`	string	-	Provide either `fleet_id` or `target_topic`.
`target_topic`	string	-	If set, publishes there instead of the default fleet topic.
`retain`	bool	`false`	If true, publishes the layout retained.
`response_topic`	string	-	Optional response topic for replication status.
`correlation_id`	string	-	Optional request/response correlation id.

Publishes to:

- Default: `kingkiosk/fleet/{fleet_id}/layout`
- Or `target_topic` when provided.

Published payload on the fleet topic:

`{command:'apply_layout', source_device, replicated_at, fleet_id, layout, window_count}`

0.1.13.3.24.2 `subscribe_fleet`

Top-level keys:

Key	Type	Default	Notes
`fleet_id`	string	(required)	Fleet id to subscribe to.
`auto_apply`	bool	`true`	If true, applies received layouts automatically.
`response_topic`	string	-	Optional response topic for subscription status.
`correlation_id`	string	-	Optional request/response correlation id.

Subscribes to `kingkiosk/fleet/{fleet_id}/layout`.

0.1.13.3.24.3 `unsubscribe_fleet`

Top-level keys:

Key	Type	Notes
`fleet_id`	string	(required)
`response_topic`	string	Optional response topic for unsubscribe status.
`correlation_id`	string	Optional request/response correlation id.

0.1.13.3.25 Screen Schedule (`set_screen_schedule`, `list_screen_schedule`, `enable_screen_schedule`, `disable_screen_schedule`, `screen_schedule_status`, `trigger_screen_schedule`)

These commands manage a minimal time-based scheduler that applies saved screen states.

All schedule responses publish to `kingkiosk/{device}/system/response` with:

`{type:'screen_schedule', status:'ok'|'error', message, timestamp, data?}`

`set_screen_schedule` keys:

Key	Type	Default	Notes
`entries`	array	(required)	List of schedule entries.
`enabled`	bool	(no override)	Optional. If present, overrides scheduler enabled state.

Schedule entry schema:

Key	Type	Default	Notes
`id`	string	(auto)	If missing/empty, an id is auto-generated.
`screen_state` / `screenState`	string	(required)	Name of a saved screen state.
`at`	string	(required)	Local time in `HH:MM`.
`days`	array	(all days)	Optional. 1=Mon … 7=Sun.
`enabled`	bool	`true`	-

`trigger_screen_schedule` keys:

Key	Type	Notes
`id`	string	Optional. Triggers a specific entry.
`screen_state`	string	Optional. Triggers a specific state.

0.1.13.3.26 Conflict Resolution (`conflict_resolution`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	Must be `conflict_resolution`.
`action`	string	-	`get_status`, `set_strategy`, `clear`, `record_touch`.
`response_topic`	string	`kingkiosk/{device}/system/response`	-

`set_strategy` keys:

Key	Type	Notes
`strategy`	string	One of `touch_priority`, `mqtt_priority`, `last_wins`, `queue`, `merge`.
`touch_cooldown_ms`	int/string	Optional.
`mqtt_cooldown_ms`	int/string	Optional.
`log_conflicts`	bool	Optional.
`publish_notifications`	bool	Optional.

`record_touch` keys:

Key	Type	Notes
`window_id`	string	Optional. Records a touch interaction for a given window.

0.1.13.4 Map

Widget Type: `map`

0.1.13.4.1 Create/Open (system command: `open_map`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	Must be `open_map`.
`title`	string	`Map`	Window title.
`window_id`	string	`map_{timestamp}`	If omitted, a map ID is generated.
`opacity`	number	`1.0`	-
`x`, `y`, `width`, `height`	number	-	Optional geometry.
`provider`	object	-	Tile provider configuration.
`initial_camera`	object	-	Initial map view.
`interaction`	object	-	Interaction switches.
`correlation_id`	string	-	Optional tracking id for response.
`response_topic`	string	`kingkiosk/{device}/system/response`	Override response topic.

Provider config (`provider`) keys:

Key	Type	Default	Notes
`url_template`	string	`https://tile.openstreetmap.org/{z}/{x}/{y}.png`	Tile URL template.
`subdomains`	array	`[]`	Used with `{s}` placeholder; ignored for OSM template.
`headers`	object	`{}`	Extra HTTP headers for tile requests.
`attribution`	string	`(c) OpenStreetMap contributors`	Attribution text (shown on map).

Notes: - If `url_template` uses `https://{s}.tile.openstreetmap.org/...`, the `{s}` subdomain portion is stripped and `subdomains` are ignored to comply with OSM guidance. - Use `close_window` on `kingkiosk/{device_id}/system/cmd` to close the map (`window_id` == `element_id`).

Initial camera (`initial_camera`) keys:

Key	Type	Default	Notes
`lat`	number	`0.0`	Latitude.
`lon`	number	`0.0`	Longitude.
`zoom`	number	`1.0`	Zoom level.
`rotation`	number	`0.0`	Rotation in degrees (0 = north-up).

Interaction switches (`interaction`) keys:

Key	Type	Default	Notes
`touch_enabled`	bool	`true`	Enables local touch/gesture control.
`remote_enabled`	bool	`true`	Enables MQTT element commands.
`allow_user_drop_pins`	bool	`false`	Allows user taps to drop pins.

0.1.13.4.2 Element Commands Element commands are sent to `kingkiosk/{device_id}/element/{ele`
only if the map widget registers with the element router.

If `interaction.remote_enabled` is `false`, map-specific commands return an error response (except `get_state`, which is handled by the common widget mixin).

Command	Parameters	Description
`set_camera`	`camera`, optional `animate`, `duration_ms`	Move/rotate the map to a camera. `animate`/`duration_ms` are accepted but currently not used for animation.
`fit_bounds`	`bounds`, optional `padding_px`, `animate`	Fit camera to bounds. `animate` currently not used.
`configure`	`provider`, `initial_camera`, `interaction`	Update provider/interaction/camera settings.
`add_pins`	`pins`	Add pins (additive).
`set_pins`	`pins`	Replace all pins.
`update_pins`	`pins`	Partial update of existing pins by `pin_id`.
`remove_pins`	`pin_ids` or `pins`	Remove pins by id.
`clear_pins`	-	Remove all pins.
`add_text`	`text` or `texts`	Add text overlays (additive).
`update_text`	`text` or `texts`	Partial update of existing text overlays.
`remove_text`	`text_ids` or `text/texts`	Remove text overlays by id.
`clear_text`	-	Remove all text overlays.
`get_state`	-	Common widget command: returns current state.

Camera schema (`camera`):

Key	Type	Notes
`lat`	number	Latitude.
`lon`	number	Longitude.
`zoom`	number	Zoom level.
`rotation`	number	Rotation in degrees.

Bounds schema (`bounds`):

Key	Type	Notes
`sw`	object	South-west corner: `{ "lat": ..., "lon": ... }`.
`ne`	object	North-east corner: `{ "lat": ..., "lon": ... }`.

Pin schema (`pins`):

Key	Type	Notes
`pin_id`	string	Required identifier.
`lat`, `lon`	number	Required coordinates.
`icon`	object	`{ "type": "url|asset|base64|default", "value": "...", "content_type": "image/png" }`.
`size_px`	object	`{ "w": 32, "h": 32 }` (default 32x32).
`anchor`	object	`{ "x": 0.5, "y": 1.0 }` (normalized).
`z_index`	int	Render order.
`opacity`	number	`0.0` to `1.0`.
`interactive`	bool	Defaults to `true`.
`metadata`	object	Arbitrary JSON metadata.

Text overlay schema (`text`/`texts`):

Key	Type	Notes
`text_id`	string	Required identifier.
`text`	string	Display text.
`anchor_type`	string	`geo` or `screen`.
`geo`	object	`{ "lat": ..., "lon": ... }` (for `anchor_type: geo`).
`screen`	object	`{ "x": 0.05, "y": 0.10 }` (normalized, for `anchor_type: screen`).
`z_index`	int	Render order.
`style`	object	See style keys below.
`interactive`	bool	Defaults to `false`.
`metadata`	object	Arbitrary JSON metadata.

Text style schema (`style`):

Key	Type	Default	Notes
`font_size_px`	number	`16.0`	Font size in pixels.
`font_weight`	string	`normal`	Accepts `normal`, `bold`, `w300`, `w500`, `w600`.
`color`	string	`#FFFFFFFF`	Text color.
`background_color`	string	`#00000000`	Background color.
`padding_px`	number	`4.0`	Padding around text.
`corner_radius_px`	number	`4.0`	Rounded corners.

```
{
  "type": "map",
  "element_id": "map-1",
  "widget_id": "map-1",
  "camera": { "lat": 30.2672, "lon": -97.7431, "zoom": 14.0, "rotation": 0.0 },
  "counts": { "pins": 12, "text_overlays": 2 },
  "interaction": { "touch_enabled": true, "remote_enabled": true, "allow_user_drop_pins": false
   ↪  }
}
```

0.1.13.4.3 State fields (published on element state topic)

0.1.13.4.4 Map Events

Map-specific events are published on `kingkiosk/{device_id}/element/{elemen`

Event	Payload Notes
`camera_changed`	Includes `camera`, `source` (`touch` or `remote`), and `phase` (`start`, `change`, `end`).
`pin_selected`	Includes `pin_id`, `lat`, `lon`, and `metadata`.
`pin_dropped`	Emitted when `allow_user_drop_pins` is enabled and the user taps the map.
`text_selected`	Includes `text_id`, `text`, and `metadata` (only if `interactive` is true).

0.1.13.5 Canvas

Widget Type: `canvas`

0.1.13.5.1 Create/Open (system command: `open_canvas`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	Must be `open_canvas`.

Key	Type	Default	Notes
`title`	string	`Canvas`	Window title.
`window_id`	string	`canvas_{timestamp}`	If omitted, an ID is generated.
`opacity`	number	`1.0`	-
`x`, `y`, `width`, `height`	number	-	Optional geometry.
`doc`	object	-	Canvas document (see below).
`interaction`	object	-	Interaction settings (see below). Overrides `doc.interaction` if both provided.
`persist`	object	-	Persistence settings (see below).
`background`	object	-	Canvas background settings (see below). Overrides `doc.background` if both provided.
`grid`	object	-	Grid settings (see below). Overrides `doc.grid` if both provided.
`viewport`	object	-	Viewport settings (see below). Overrides `doc.viewport` if both provided.
`correlation_id`	string	-	Optional tracking id for response.
`response_topic`	string	`kingkiosk/{device}/system/response`	Override response topic.

Notes: - `window_id` is also used as `element_id`/`widget_id` for element commands. - Close with `close_window` on `kingkiosk/{device_id}/system/cmd` using `window_id`. - The canvas document schema is versioned by `doc.spec` (currently `kingkiosk.canvas.v1`).

0.1.13.5.2 Canvas Document (doc)

This widget uses a single **document** that contains the full scene (objects + connections) and the view/config state.

`doc` keys:

Key	Type	Default	Notes
`spec`	string	`kingkiosk.canvas.v1`	Schema identifier.
`canvas_id`	string	-	Optional application-level id.
`meta`	object	-	Arbitrary metadata.
`background`	object	-	Background config (see below).
`grid`	object	-	Grid config (see below).
`viewport`	object	-	Viewport config (see below).
`interaction`	object	-	Interaction config (see below).
`objects`	array	`[]`	Canvas objects (see below).
`connections`	array	`[]`	Connections between objects (see below).
`resources`	object	-	Arbitrary JSON resource registry (currently stored/pass-through).

Background schema (`background`):

Key	Type	Default	Notes
`color`	string	`#00000000`	#RRGGBB or #AARRGGBB.
`blur_px`	number	`0.0`	Backdrop blur for the whole canvas.
`image`	object	-	`{ "url": "..." }` or `{ "asset_id": "..." }`.

Grid schema (`grid`):

Key	Type	Default	Notes
`enabled`	bool	`false`	-
`size_px`	number	`20.0`	Grid step size in pixels.
`color`	string	`#22FFFFFF`	Grid line color.

Viewport schema (`viewport`):

Key	Type	Default	Notes
`zoom`	number	`1.0`	-
`pan_x`	number	`0.0`	-
`pan_y`	number	`0.0`	-
`min_zoom`	number	-	Optional clamp.
`max_zoom`	number	-	Optional clamp.

Interaction schema (`interaction`):

Key	Type	Default	Notes
`touch_enabled`	bool	`true`	Enables local touch/drag interactions.
`remote_enabled`	bool	`true`	Enables element-scoped MQTT commands (except `get_state`).
`edit_mode`	bool	`false`	Enables local dragging (with `touch_enabled`).
`snap_to_grid`	bool	`false`	Enables snapping while dragging.
`snap_px`	number	-	If omitted, uses `grid.size_px`.
`show_ports_in_view`	bool	`false`	Shows port markers on objects.
`allow_add_objects`	bool	`false`	Reserved for future in-app add flows (currently not enforced for MQTT).
`allow_add_connections`	bool	`false`	Reserved for future in-app add flows (currently not enforced for MQTT).

Persist schema (`persist`):

Key	Type	Default	Notes
`enabled`	bool	`true`	Enables persistence in local storage.

Key	Type	Default	Notes
`key`	string	`{device}:{window_id}`	Storage key (falls back to `{window_id}` if device id unavailable).
`restore_on_start`	bool	`true`	Restores the last saved document on start.
`publish_on_restore`	bool	`true`	Publishes retained state after restore.

0.1.13.5.3 Objects (`objects`) Objects are positioned in **canvas coordinates** (affected by `viewport.zoom/pan` in the renderer).

Common object keys:

Key	Type	Default	Notes
`object_id`	string	(required)	Unique id.
`type`	string	`node`	One of: `node`, `text`, `shape`, `image`, `embed`, `group`.
`x`, `y`	number	`0.0`	Top-left position.
`width`, `height`	number	`0.0`	Object size.
`z_index`	int	`0`	Render order (used when no explicit order is set).
`visible`	bool	`true`	-
`locked`	bool	`false`	Prevents local dragging.
`rotation_deg`	number	`0.0`	Rotation around object center.
`style`	object	`{}`	Styling keys (see below).

Key	Type	Default	Notes
`ports`	array	`[]`	Optional port list (see below).
`mqtt`	object	`{}`	Optional tap → publish behavior (see below).
`metadata`	object	`{}`	Arbitrary metadata.

`style` keys currently used by the renderer: - `color`: object background color (default: transparent for `text`/`shape`, dark for others) - `border_color`: outline color - `border_width`: outline width - `fill_color`: (for `shape`) fill color - `stroke_color`: (for `shape`) stroke color

Tap → publish (`mqtt`) behavior:

If `mqtt.publish` is set, a local tap publishes to the specified topic.

Key	Type	Notes
`publish.topic`	string	Target topic.
`publish.payload`	any	JSON object → publish as JSON; otherwise published as a string.
`publish.retain`	bool	Retained publish when `true`.

Ports (`ports`) schema:

Key	Type	Default	Notes
`port_id`	string	(required)	Port identifier.
`pos`	object	-	Either `{ "nx": 0..1, "ny": 0..1 }` or `{ "edge": "north|south|east|west", "t": 0..1 }`.
`kind`	string	-	Optional label/typing (stored only).

Key	Type	Default	Notes
`label`	string	-	Optional label (stored only).
`style`	object	-	Currently uses `style.color` for marker fill.

Notes: - If `ports` is omitted/empty, the renderer uses implicit ports: `north`, `south`, `west`, `east`, `center`. - For connections, if `port_id` is omitted (or does not match any declared port), the renderer falls back to the object center (and also supports the implicit port ids above).

Type-specific keys (stored on the object and used by the renderer):

- `type: node`
  - `label` (string), `subtitle` (string)
  - `icon` (object): `{ "set": "material|sf", "name": "...", "color": "#AARRGGBB", "size_px": 24 }`
- `type: text`
  - `text` (string)
  - `font_size_px` (number, default 16)
  - `font_weight` (string: `normal|bold|w300|w500|w600`)
  - `color` (string: #RRGGBB or #AARRGGBB)
  - `align` (string: `left|center|right`)
- `type: shape`
  - `shape` (string: `rect|round_rect|circle|line`)
  - `corner_radius_px` (number, default 8)
  - `stroke_width_px` (number, default 2)
  - Uses `style.fill_color` and `style.stroke_color`
- `type: image`
  - `source` (object): `{ "url": "..." }` or `{ "asset_id": "..." }`
  - `fit` (string: `contain|cover|fill`, default `contain`)
  - `opacity` (number, default `1.0`)
- `type: embed`

    - embed (object): `{ "widget_type": "gauge|chart|mqtt_button|mqtt_action_status", "id": "...", "config": { ... } }`
    - `config` is passed through to the embedded widget; see the corresponding widget sections in this document.

- `type: group`
    - No additional keys (renders as a transparent rectangle with a border).

0.1.13.5.4 Connections (`connections`)

Common connection keys:

Key	Type	Default	Notes
`connection_id`	string	(required)	Unique id.
`from`	object	(required)	`{ "object_id": "...", "port_id": "..." }` (`port_id` optional).
`to`	object	(required)	`{ "object_id": "...", "port_id": "..." }` (`port_id` optional).
`style`	object	-	Connection style (see below).
`route`	object	-	Connection routing (see below).
`mqtt`	object	`{}`	Reserved (stored only).
`metadata`	object	`{}`	Arbitrary metadata.

Connection style schema (`style`):

Key	Type	Default	Notes
`color`	string	`#FFFFFFFF`	Line color.
`width_px`	number	`2.0`	Line width.

Key	Type	Default	Notes
`dash`	array	`[]`	Dash pattern list (alternating draw/gap lengths).
`arrow`	string	`none`	`none`, `end`, `both`.
`opacity`	number	`1.0`	-
`animated`	bool	`false`	Renders an animated flow (moving dashes/highlight).
`animation_speed`	int	`3`	Speed 1-5 (1=very slow at 0.3x, 2=0.6x, 3=1.0x, 4=1.5x, 5=very fast at 2.0x). Clamped to [1,5].

Connection route schema (`route`):

Key	Type	Default	Notes
`kind`	string	`auto`	If `manual`, uses `points`.
`mode`	string	`orthogonal`	`straight`, `curved`, `orthogonal`.
`points`	array	`[]`	For `kind: manual`: list of `{ "x": ..., "y": ... }`.

0.1.13.5.5 Element Commands Element commands are sent to `kingkiosk/{device_id}/element/{ele` after the widget is created/registered.

If `interaction.remote_enabled` is `false`, canvas-specific commands return an error response (except `get_state`, which is handled by the common widget mixin).

Command	Parameters	Description
`configure`	optional `interaction`, `background`, `grid`, `viewport`	Updates view/config settings (does not change objects/connections).
`set_document`	`doc`	Replace entire document (objects + connections + configs).
`apply_patch`	optional `base_rev`, `ops[]`	Apply a small patch to the document (see patch format below).
`add_objects`	`objects`	Add objects (additive).
`set_objects`	`objects`	Replace all objects.
`update_objects`	`objects`	Deep-merge object updates by `object_id`.
`remove_objects`	`object_ids`	Remove objects by id.
`clear_objects`	-	Remove all objects.
`add_connections`	`connections`	Add connections (additive).
`set_connections`	`connections`	Replace all connections.
`update_connections`	`connections`	Deep-merge connection updates by `connection_id`.
`remove_connections`	`connection_ids`	Remove connections by id.
`clear_connections`	-	Remove all connections.
`get_state`	-	Common widget command: returns current state.

Patch format (`apply_patch`):

Key	Type	Notes
base_rev	int	Optional optimistic concurrency; if provided and mismatched, returns {status:"error", code:"rev_mismatch", current_rev} and publishes event: patch_rejected.
ops	array	List of operations (see below).
correlation_id	string	Optional; echoed in events.

Patch operations (ops[]) are processed in order:

Key	Type	Notes
op	string	set, merge, delete, reorder.
path	string	JSON-pointer-like path, e.g. /objects/byId/{object_id}/x.
value	any	For set/merge.
ids	array	For reorder only.

Supported paths: - /objects/byId/{object_id} or /objects/byId/{object_id}/{field...} - /connections/byId/{connection_id} or /connections/byId/{connection_id}/{field...} - /objects with op: reorder and ids: ["obj1","obj2",...] sets render order - /connections with op: reorder and ids: ["c1","c2",...] sets render order

```
{
  "type": "canvas",
  "element_id": "canvas-1",
  "widget_id": "canvas-1",
  "rev": 12,
  "counts": { "objects": 7, "connections": 3 },
  "interaction": { "touch_enabled": true, "remote_enabled": true, "edit_mode": false },
  "doc": { "spec": "kingkiosk.canvas.v1", "objects": [], "connections": [] },
  "last_error": { "message": "..." }
}
```

0.1.13.5.6 State fields (published on element state topic)

0.1.13.5.7 Canvas Events

Canvas events are published on `kingkiosk/{device_id}/element/{element`

Event	Description	Fields
`doc_changed`	Document changed	`rev`, `source`
`patch_applied`	Patch accepted	`rev`, optional `correlation_id`
`patch_rejected`	Patch rejected	`code`, `current_rev`, optional `correlation_id`
`object_moved`	Local drag interaction	`object_id`, `phase` (`start`|`change`|`end`), `x`, `y`, `source` (`touch`)

0.1.13.6 Animated Text

Widget Type: `animatedText`

0.1.13.6.1 Create/Open (system command: `open_animated_text`)

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	Must be `open_animated_text`.
`title`	string	`Animated Text`	Window title.
`window_id`	string	`animated_text_{timestamp}`	If omitted, an ID is generated.

Key	Type	Default	Notes
`opacity`	number	`1.0`	-
`x`, `y`, `width`, `height`	number	-	Optional geometry.
`spec`	object	-	Full animated text spec (see below).
`text`	string	-	Convenience: text content if you aren't sending `spec`.
`segments`	array	-	Optional rich text segments (overrides `text`).
`layout`	object	-	Layout config.
`style`	object	-	Text styling.
`tokenization`	object	-	Tokenization config.
`timeline`	object	-	Timeline config.
`effects`	array	-	Effect list.
`audio`	object	-	Audio config.
`lod`	object	-	LOD config.
`mqtt`	object	-	Stored in state; not used for auto-subscribe in current implementation.
`animation`	object	-	Accepts `{ preset: "..." }` as shorthand.
`preset`	string	-	Preset name (see below).
`id`	string	-	Optional spec id.
`correlation_id`	string	-	Optional tracking id for response.

Key	Type	Default	Notes
`response_topic`	string	`kingkiosk/{device}/system/response`	Override response topic.

Notes: - `window_id` is also used as `element_id`/`widget_id` for element commands. - You can provide either a full `spec` object or top-level keys (they are merged into the spec). - Close with `close_window` on `kingkiosk/{device_id}/system/cmd` using `window_id`.

0.1.13.6.2 Animated Text Spec (`spec`) This widget's core configuration is a single `spec` object. The system `open_animated_text` command accepts either: - a full `spec` object (in `spec`), plus optional top-level overrides, or - top-level spec fields without `spec`.

Spec keys:

Key	Type	Default	Notes
`id`	string	`{window_id}`	If omitted, falls back to the window id.
`text`	string	`""`	Plain text content (ignored when `segments` is non-empty).
`segments`	array	`[]`	Rich segments (see below).
`layout`	object	-	Layout config (see below).
`style`	object	-	Base style (see below).
`tokenization`	object	-	Tokenization config (see below).
`timeline`	object	-	Timeline config (see below).
`effects`	array	`[]`	Effect stack (see below).
`audio`	object	-	Optional audio config (see below).

Key	Type	Default	Notes
`lod`	object	-	Optional LOD config (see below).
`mqtt`	object	-	Stored in state; not auto-subscribed in current implementation.
`preset`	string	-	Preset name (applied at create time or via `set_preset`).

Segments schema (`segments[]`):

Key	Type	Default	Notes
`text`	string	`""`	Segment content.
`style`	object	`{}`	Partial style patch applied only for this segment (same keys as `style`).

Layout schema (`layout`):

Key	Type	Default	Notes
`wrap`	string	`word`	Controls line wrapping; set to none to disable wrapping.
`maxLines` / `max_lines`	int	-	Max visible lines.
`alignment`	string	`left`	`left`, `center`, `right`.
`lineHeight` / `line_height`	number	`1.0`	-

Key	Type	Default	Notes
`letterSpacing` / `letter_spacing`	number	`0.0`	-
`wordSpacing` / `word_spacing`	number	`0.0`	-
`overflow`	string	`clip`	Stored only; renderer currently uses clipping.

Style schema (`style`):

Key	Type	Default	Notes
`fontFamily` / `font_family`	string	-	-
`fontSize` / `font_size`	number	`32.0`	-
`weight`	string	`600`	Passed through to Flutter's `FontWeight` parsing.
`color`	string	`#FFFFFFFF`	#RRGGBB or #AARRGGBB.
`stroke`	object	-	See below.
`shadow`	object	-	See below.

Stroke schema (`style.stroke`):

Key	Type	Default	Notes
`width`	number	`0.0`	-
`color`	string	`#00000000`	Stroke color.

Shadow schema (`style.shadow`):

Key	Type	Default	Notes
`blur`	number	`0.0`	Blur radius.
`dx`	number	`0.0`	X offset.
`dy`	number	`0.0`	Y offset.
`color`	string	`#00000000`	Shadow color.

Tokenization schema (`tokenization`):

Key	Type	Default	Notes
`mode`	string	`grapheme`	`grapheme`, `word`, `line`, `segment`.
`animateBy` / `animate_by`	string	`character`	Stored only (not currently used).
`preserveSpaces` / `pre-serve_spaces`	bool	`false`	When true, spaces become tokens and can animate.
`rtl`	string	`auto`	Stored only (not currently used).

Timeline schema (`timeline`):

Key	Type	Default	Notes
`mode`	string	`once`	`once` or `loop`.
`loopCount` / `loop_count`	int	`0`	Stored only (not currently enforced).
`direction`	string	`forward`	Stored only (not currently enforced).
`delayMs` / `delay_ms`	int	`0`	Delay before effects start.

Key	Type	Default	Notes
`repeatDelayMs` / `re-peat_delay_ms`	int	`0`	Delay between loops.

Effects schema (`effects[]`):

Common keys:

Key	Type	Default	Notes
`type`	string	`fade`	See supported types below.
`durationMs` / `duration_ms`	int	`500`	-
`startOffsetMs` / `start_offset_ms` / `startOffset`	int	`0`	-
`easing`	string	`linear`	Supported: `linear`, `easeOutCubic`, `easeInOutCubic`, `easeOutBack`.
`stagger`	object	-	`{ "by": "tokenIndex", "eachMs": 0 }` (alias: `each_ms`). Only `eachMs` is used by the renderer.

Supported effect types: - `fade`: uses numeric `from`/`to` (`effect.from.opacity`/`effect.to.opacity` also accepted). - `typewriter`: treated like `fade` (typically paired with token staggering). - `slide`: uses `from: {x,y}`, `to: {x,y}` in pixels. - `scale`: uses numeric `from`/`to`. - `colorize`: set `mode: "solidLerp"` with `from`/`to` colors, or set `mode: "paletteCycle"` with `colors:`

`["#...","#..."]` for palette interpolation. - `marquee`: scrolls the whole text group; uses `speedPxPerSec` (default `90`), `gapPx` (default `48`), and `direction` (`left`/`right`).

Audio schema (`audio`):

Key	Type	Default	Notes
`enabled`	bool	`false`	-
`mode`	string	`perCharacter`	Stored only; the renderer emits at most one sound per token step.
`sound`	string	`notification`	AudioService key.
`volume`	number	`0.2`	Stored only; AudioService implementation decides final mix.
`rateLimit`	object	-	`{ "maxPerSecond": 12, "burst": 4 }` (alias: `max_per_second` for `maxPerSecond`).

LOD schema (`lod`):

Key	Type	Default	Notes
`maxGraphemes` / `max_graphemes`	int	`0`	If > 0 and the text exceeds this, the renderer falls back.
`fallbackAnimateBy` / `fall-back_animate_by`	string	`word`	Used as the fallback tokenization mode.

MQTT schema (`mqtt`):

Key	Type	Notes
`subscribe.textTopic` / `subscribe.text_topic`	string	Stored only.
`subscribe.styleTopic` / `subscribe.style_topic`	string	Stored only.
`subscribe.effectTopic` / `subscribe.effect_topic`	string	Stored only.
`subscribe.triggerTopic` / `subscribe.trigger_topic`	string	Stored only.
`publish.eventsTopic` / `publish.events_topic`	string	Stored only.

Background decoration keys (not part of `spec`, stored on the tile metadata):

Key	Type	Default	Notes
`background_mode`	string	`color`	`transparent`, `color`, `gradient`, `image`.
`background_color`	string	`#000000`	Base color for `color`/`gradient`.
`background_opacity`	number	`0.8`	Clamped to `0.0`–`1.0`.
`background_image_url`	string	-	Used when `background_mode: image`.

0.1.13.6.3 Element Commands Element commands are sent to `kingkiosk/{device_id}/element/{ele`
after the widget is created/registered.

Command	Parameters	Description
`configure`	any spec fields	Replaces the entire spec from the payload (missing fields fall back to defaults). Prefer `set_*` commands for targeted updates.
`set_text`	`text`	Replace text content.
`set_segments`	`segments`	Replace segment list.
`set_style`	`style`	Replace text style.
`set_layout`	`layout`	Replace layout config.
`set_tokenization`	`tokenization`	Replace tokenization config.
`set_timeline`	`timeline`	Replace timeline config.
`set_effects`	`effects`	Replace effects list.
`set_audio`	`audio`	Replace audio config.
`set_preset`	`preset`	Apply a preset (see below).
`trigger`	optional `source`	Restarts animation sequence and publishes `event: play`.

Preset names currently supported: - `typewriterShimmer` - `neonPulse` - `bounceCascade` - `alertFlash` - `tickerMarquee`

0.1.13.6.4 State fields (published on element state topic)

Field	Type	Notes
`type`	string	`animatedText`
`element_id`	string	Same as window_id.
`widget_id`	string	Same as window_id.
`spec`	object	Full animated text spec currently applied.

0.1.13.6.5 Animated Text Events Standard widget lifecycle events apply (`created`, `closed`, `error`). This widget also publishes:

Event	Description	Fields
doc_changed	Spec updated via MQTT	source
play	Triggered animation restart	source

0.1.13.7 Clock

Widget Type: clock

0.1.13.7.1 Create/Open (system command: open_clock)

Top-level keys:

Key	Type	Default	Notes
command	string	(required)	Must be open_clock.
title	string	Analog Clock	Window title.
window_id	string	(auto)	If provided, creates with that ID.
opacity	number	1.0	-
x, y, width, height	number	-	Optional geometry.
mode	string	-	analog or digital.
image_url	string	-	Network image URL.
theme	string	-	auto, light, dark.
show_numbers	bool/string	-	Truthy values accepted.
show_second_hand	bool/string	-	Truthy values accepted.

0.1.13.7.2 Element Commands

This widget may optionally support element-scoped commands on kingkiosk/{device_id}/element/{element_id}/cmd **only if it registers** a handler with MqttWidgetRouter.registerWidget(...).

Clock configuration keys (used by open_clock):

Key	Type	Notes
mode	string	analog or digital.

Key	Type	Notes
`image_url`	string	Sets `network_image_url`.
`show_numbers`	bool/string	Truthy values accepted.
`show_second_hand`	bool/string	Truthy values accepted.
`theme`	string	-
`background_mode`	string	One of: `transparent`, `gradient`, `color`, `image`.
`background_color`	string	#RRGGBB or #AARRGGBB.
`background_opacity`	number/string	Clamped to `0.0–1.0`.
`background_image_url`	string	-
`visible`	bool	-

0.1.13.7.3 Element Commands (topic: `kingkiosk/{device_id}/element/{element_id}/cmd`)

Clock currently implements element commands via the per-element router.

Command	Parameters	Description
`set_mode`	`mode`	Set display mode.
`toggle_mode`	-	Toggle analog/digital.
`configure`	see `configure` keys above	Configure appearance.
`minimize`	-	Minimize/hide.
`maximize` / `restore`	-	Restore.
`close`	-	Close.

```
{
  "type": "clock",
  "element_id": "clock-1",
  "widget_id": "clock-1",
  "mode": "analog",
  "visible": true,
  "minimized": false,
  "show_numbers": true,
  "show_second_hand": true,
  "theme": "auto",
  "background_mode": "transparent",
  "background_opacity": 0.6
}
```

0.1.13.7.4 State fields (published on element + widget state topics)

0.1.13.8 Weather (OpenWeather)

Widget Type: `weather`

0.1.13.8.1 Create/Open (system command: `open_weather_client`)

Top-level keys used:

Key	Type	Default	Notes
`command`	string	(required)	Must be `open_weather_client`.
`name`	string	`Weather`	Window title.
`window_id`	string	(auto)	-
`opacity`	number	`1.0`	-
`x`, `y`, `width`, `height`	number	-	Optional geometry.
`api_key`	string	-	OpenWeather API key.
`location`	string	-	City name (alternative to coordinates).
`units`	string	`imperial`	`metric`, `imperial`, or `standard`.
`language`	string	`en`	Controller maps: `en`, `de`, `fr`, `es`, `it`; others default to `en`.

Key	Type	Default	Notes
`show_forecast`	bool/string/int	`false`	Accepts `true`/`false`, `"true"`/`"false"`, `"yes"`/`"on"`, `1`/`0`.
`auto_refresh`	bool/int	`true`	If int > 0, treated as refresh interval seconds.
`refresh_interval`	int/string	`3600` (default)	Parsed as int; controller fallback is `300` if parse fails.

0.1.13.8.2 Element Commands

Element-scoped commands on `kingkiosk/{device_id}/element/{ele`

Command	Parameters	Description
`configure`	Any config keys below	Update widget configuration.
`refresh`	-	Force a weather data refresh.
`toggle_forecast`	-	Toggle forecast panel visibility.
`set_location`	`location` (string)	Change weather location.
`hide`	-	Hide the widget.
`show`	-	Show the widget.

Weather configuration keys (used by `open_weather_client` and `configure`):

Key	Type	Notes
`api_key`	string	Required for fetching.
`location`	string	City name.
`latitude`, `longitude`	number/string	Coordinate alternative.

Key	Type	Notes
`units`	string	`metric`, `imperial`, `standard` (default: `imperial`).
`language`	string	Mapped set; defaults to en.
`show_forecast`	bool/string/int	Accepts `true`/`false`, `"true"`/`"false"`, `1`/`0`.
`auto_refresh`	bool/int	If int > 0, also sets refresh interval seconds.
`refresh_interval`	int/string	Parsed int; default fallback `300`.
`allow_bad_cert`	bool/string/int	DEV ONLY. Accepts `true`/`false`, `"true"`/`"false"`, `"yes"`/`"on"`, `1`/`0`.

Note: `latitude`, `longitude`, and `allow_bad_cert` are only processed via element-level `configure` commands; they are **not** passed through the `open_weather_client` system command handler.

0.1.13.9 Alarmo

Widget Type: `alarmo`

0.1.13.9.1 Create/Open (system command: `alarmo` / `alarmo_widget`) Top-level keys used:

Key	Type	Default	Notes
`command`	string	(required)	`alarmo` or `alarmo_widget`.
`name`	string	`Alarmo`	Window title.
`window_id`	string	(auto)	-

Key	Type	Default	Notes
`opacity`	number	`1.0`	-
`x`, `y`, `width`, `height`	number	-	Optional geometry.
`entity`	string	-	Home Assistant entity id.
`require_code`	bool	`true`	Legacy shorthand — sets both `require_code_to_arm` **and** `require_code_to_disarm`.
`require_code_to_arm`	bool	`true`	Whether a PIN is required for **arming**. Overrides `require_code`.
`require_code_to_disarm`	bool	`true`	Whether a PIN is required for **disarming**. Overrides `require_code`.
`code_required_modes`	object	`{}`	Per-mode override map, e.g. `{"away": true, "home": false}`. Keys are mode names (away, home, `night`, `vacation`, `custom`). When a mode is present in this map its value takes precedence over `require_code_to_arm`.
`code_length`	int	4	PIN digit count.

Key	Type	Default	Notes
`mqtt_base_topic`	string	`"alarmo"`	Used by controller to construct state/command/event topics.
`state_topic` / `command_topic` / `event_topic`	string	legacy	Accepted by create handler for backward compatibility.
`area`	string	optional	For multi-area Alarmo. Slug format (lowercase, underscores).
`available_modes`	array	`["armed_away"]`	Strings like `armed_away`, `armed_home`, `armed_night`, `armed_vacation`, `armed_custom_bypass`.
`auto_recovery`	bool	`true`	Enables auto recovery on arm failure.
`force`	bool	`false`	Default state for "force arm" (bypass open sensors). Can be toggled in the UI.
`skip_delay`	bool	`false`	Default state for "skip exit delay". Can be toggled in the UI.

0.1.13.9.2 Element Commands Element-scoped commands on `kingkiosk/{device_id}/element/{ele`

Command	Parameters	Description
`configure`	Any config keys below	Update widget configuration.
`arm`	`mode` (string, e.g. away), optional `code` (string), optional `force` (bool), optional `skip_delay` (bool)	Arm the alarm. `force` bypasses open sensors; `skip_delay` skips exit delay.
`disarm`	optional `code` (string)	Disarm the alarm.
`set_force` / `force_arm`	`value` (bool)	Enable/disable force arm toggle.
`set_skip_delay` / `skip_delay`	`value` (bool)	Enable/disable skip delay toggle.
`minimize`	—	Minimize the window.
`maximize` / `restore`	—	Restore the window.
`close`	—	Close the window.

Alarmo configuration keys (used by `alarmo` / `alarmo_widget` create/open):

Key	Type	Notes
`entity`	string	Home Assistant entity id.
`require_code`	bool	Legacy shorthand — sets both arm and disarm code requirement.
`require_code_to_arm`	bool	Overrides `require_code` for arming.
`require_code_to_disarm`	bool	Overrides `require_code` for disarming.
`code_required_modes`	object	Per-mode override map, e.g. `{"away": true, "home": false}`.
`code_length`	int	PIN digit count (default 4).
`mqtt_base_topic`	string	Base topic (default `alarmo`).

Key	Type	Notes
`area`	string	Area name for multi-area setups.
`auto_recovery`	bool	Enable auto recovery on failure.
`available_modes`	array	Strings starting with `armed_` are parsed into allowed arm modes.
`force`	bool	Default force arm state (bypass open sensors).
`skip_delay`	bool	Default skip exit delay state.

0.1.13.10 MQTT Button (MQTT Action Status)

Widget Type: `mqtt_action_status` (preferred system command name: `mqtt_button`)

0.1.13.10.1 Create/Configure (system command: `mqtt_button` / `mqtt_action_status` / `action_status`)

This handler supports "create on configure": if `action == configure` and `window_id` does not exist, it will create a new tile.

Top-level keys used:

Key	Type	Default	Notes
`command`	string	(required)	`mqtt_button` / `mqtt_action_status` / `action_status`.
`action`	string	`trigger`	Common values: `configure` (alias: `update_config`), `trigger` (alias: `execute`), `publish`/`send`, `toggle`, `set_status`.

Key	Type	Default	Notes
`window_id`	string	-	Required for window-scoped actions; optional for generic publish.
`topic`	string	-	Convenience alias for `publish_topic`.
`payload`	any	-	Convenience alias for `publish_payload`.
`publish_topic` / `publishTopic`	string	-	MQTT topic to publish when triggered.
`publish_payload` / `publishPayload`	object/any	-	MQTT payload published.
`subscription_topic`	string	-	Topic to subscribe to for status.
`label`	string	-	UI label.
`mode` / `display_mode`	string	-	`toggle`/`switch` or `icon_button`/`button`.
`icon_on`, `icon_off`	string	-	Icon names (controller maps these to icons).
`color_on`, `color_off`	string	-	Named color string (e.g., `red`, `green`, `blue`, `grey`), hex string (`#RRGGBB`, `#AARRGGBB`), or `0x` prefixed string.
`size`	number	`48.0`	Icon size. Clamped to [16.0, 128.0].

Key	Type	Default	Notes
`confirm`	bool	`false`	If true, publishes an acknowledgement to `kingkiosk/{device}/system/`

0.1.13.10.2 Element Commands

This widget may optionally support element-scoped commands on `kingkiosk/{device_id}/element/{element_id}/cmd` **only if it registers** a handler with `MqttWidgetRouter.registerWidget(...)`.

0.1.13.11 Charts

Charts are managed via the chart command set.

Rendering defaults now use a dark panel, animated transitions (400ms), gradient fills, rounded bars, styled tooltips, and a larger donut-style pie center.

0.1.13.11.1 Visual Defaults

Element	Default
Panel background	`#161A24` → `#0F1117` vertical gradient
Primary color	`#6366F1` (indigo)
Accent color	`#06B6D4` (cyan)
Grid lines	Dashed horizontal lines (`#4B5568` with alpha)
Pie palette	Indigo, cyan, amber, red, violet, emerald, orange
Pie center radius	`48`
Bar corners	Rounded top corners (`6px`)

0.1.13.11.2 `create_chart`

Key	Type	Default	Notes
`chart_id`	string	(required)	Identifier.
`window_id`	string	`chart_id`	-
`chart_type`	string	-	Preferred. Alias: `type`.
`type`	string	-	Alias for `chart_type`.
`max_points`	int	`60`	Max points retained.
`mqtt_topic_prefix`	string	`kingkiosk`	Used for publish/subscribe topic construction.
`title`	string	-	-
`opacity`, `x`, `y`, `width`, `height`	number	-	Optional geometry.

0.1.13.11.3 `append_chart_data`

Key	Type	Notes
`chart_id`	string	Required.
`value`	number	Required.
`mqtt_topic_prefix`	string	Optional.

0.1.13.11.4 `replace_chart_data`

Key	Type	Notes
`chart_id`	string	Required.
`values`	array	Required.
`mqtt_topic_prefix`	string	Optional.

0.1.13.11.5 `update_pie_chart`

Key	Type	Notes
`chart_id`	string	Required.
`slices`	array	Required.
`mqtt_topic_prefix`	string	Optional (default `kingkiosk`).

Slice schema:

Key	Type	Notes
`value`	number	Required.
`label`	string	Optional.
`color`	string/int	Optional. Accepts #RRGGBB, #AARRGGBB, or int ARGB.

0.1.13.11.6 `configure_chart`

Key	Type	Notes
`chart_id`	string	Required.
`config`	object	Required.
`mqtt_topic_prefix`	string	Optional (default `kingkiosk`).

Known `configure_chart` command config keys:

Key	Type	Notes
`type`	string	`bar`, `pie`, `line`.
`primaryColor`	string/int	Color.

Note: `configure_chart` (system command) currently applies `type` and `primaryColor`. For `title` and full visual options, publish directly to the chart config topic below.

0.1.13.11.7 `reset_chart`

Key	Type	Notes
`chart_id`	string	Required.
`mqtt_topic_prefix`	string	Optional (default `kingkiosk`).

0.1.13.11.8 `delete_chart`

Key	Type	Notes
`chart_id`	string	Required.
`window_id`	string	Optional (defaults to `chart_id`).
`mqtt_topic_prefix`	string	Optional (default `kingkiosk`).

0.1.13.11.9 `list_charts`

No parameters. Returns a list of all chart tiles.

0.1.13.11.10 Direct MQTT Chart Topics (Controller-Level)

These topics are also supported by the chart controller and expose the full config surface.

Topic	Direction	Purpose
`{prefix}/chart/{chartId}/config`	Subscribe	Apply chart configuration
`{prefix}/chart/{chartId}/data`	Subscribe	Append/replace time-series data
`{prefix}/chart/{chartId}/pie`	Subscribe	Replace pie slices
`{prefix}/chart/{chartId}/reset`	Subscribe	Clear chart data

Config topic keys (`{prefix}/chart/{chartId}/config`):

Key	Type	Notes
`type`	string	`line`, `bar`, `pie`
`title` / `chart_title`	string	In-panel chart title

Key	Type	Notes
`primaryColor`	string/int	Primary chart color
`showGrid`	bool	Show dashed horizontal grid lines
`showAxes`	bool	Show axis labels
`showDots`	bool	Line chart dots
`curvedLines`	bool	Curved line interpolation
`fillBelow`	bool	Gradient fill under line
`lineStrokeWidth`	number	Line thickness
`barWidth`	number	Bar width
`minY`, `maxY`	number	Fixed y-axis bounds
`showLabels`	bool	Pie labels
`pieCenterSpaceRadius`	number	Donut center radius
`pieSectionsSpace`	number	Gap between pie slices
`pieRadius`	number	Pie slice radius

Data topic payloads (`{prefix}/chart/{chartId}/data`):

- Append: `{ "append": 72.5 }`
- Replace: `{ "replace": [68, 70, 72] }`
- Replace (array form): `[68, 70, 72]`

0.1.13.12 MQTT Gauges

The Gauge widget provides visual representation of values within a range, supporting multiple display styles and interactive controls. Designed for kiosk applications like thermostats, meters, and dashboards.

Rendering is now fully custom-painted (no third-party gauge package), with animated value transitions (600ms, `easeOutCubic`) for both primary values and pointers.

0.1.13.12.1 Core Features

- **Display Styles**: linear, circular/radial, semicircular, thermostat
- **Interactive Mode**: User can adjust values via remote/touch/keyboard
- **Locked Mode**: Display-only, values controlled via MQTT
- **Multi-Pointer Support**: Multiple indicators on single gauge (e.g., current temp + setpoints)
- **Thresholds/Zones**: Color zones based on value ranges
- **MQTT Bidirectional**: Subscribes to value updates, publishes user changes
- **Thermostat Dual Setpoint Mode**: Heat/cool range arc with current pointer and mode indicator (`Heating` / `Cooling` / `Idle`)
- **Safe Thermostat Interaction**: In multi-pointer thermostat mode, setpoints update only when dragging a pointer handle (background taps do not change values)

0.1.13.12.2 Visual Defaults

Element	Default
Track background	`#2A2D35`
Value color theme	`#6366F1 → #06B6D4`
Label color	`#B0B8C8`
Track thickness	`12px`
Thermostat heat color	`#FF6B35`
Thermostat cool color	`#4FC3F7`

0.1.13.12.3 Create Gauge (`create_gauge`, `create_mqtt_gauge`)

Key	Type	Default	Notes
`gauge_id`	string	(required*)	Unique identifier for the gauge
`window_id`	string	`gauge_id`	Window/tile identifier
`title`	string	`"Gauge {gauge_id}"`	Display title

Key	Type	Default	Notes
`gauge_type`	string	`"linear"`	Style: `"linear"`, `"circular"`, `"radial"`, `"semicircular"`, `"thermostat"`
`min`	number	`0`	Minimum value
`max`	number	`100`	Maximum value
`default_value`	number	`0`	Initial value (alias: `value`)
`value`	number	`0`	Alias for `default_value`
`unit`	string	`""`	Unit label (e.g., `"°F"`, `"%"`, `"kW"`)
`interactive`	bool	`true`	Allow user to adjust value
`locked`	bool	`false`	Lock primary value (read-only display)
`step_size`	number	`null`	Increment step for user adjustments (alias: `stepSize`). If omitted/null, values are continuous (no snapping).
`decimals`	number	`0`	Decimal places to display (0-4)
`mqtt_topic_prefix`	string	`"kingkiosk"`	Base topic for pub/sub
`color_mode`	string	-	`"gradient"`, `"thresholds"`, `"solid"`, `"zones"`

Key	Type	Default	Notes
`show_min_max`	bool	`true`	Show min/max labels
`show_value`	bool	`true`	Show current value
`thresholds`	array	-	Color threshold definitions (see below)
`zones`	array	-	Color zone definitions (see below)
`config`	object	-	Additional configuration (pointers, zones)
`os_widget`	bool	`false`	Create native OS widget (Android/iOS home screen)
`mqtt_topic`	string	-	MQTT topic for OS widget to subscribe to (required if `os_widget: true`)
`json_field`	string	-	OS widget JSON extraction path (for `os_widget`). In-app multi-pointer extraction uses `pointers[].json_field`.
`opacity`, `x`, `y`, `width`, `height`	number	-	Optional geometry

Important: In-app gauges do not use top-level `subscribe_topic` / `publish_topic`. Use pointer-level MQTT fields in `pointers[]`.

Threshold Object:

Key	Type	Required	Notes
`value`	number	Yes	Threshold value
`color`	string	Yes	Hex color (e.g., `"#3498db"`)
`label`	string	No	Optional label (e.g., `"Cold"`)

Zone Object:

Key	Type	Required	Notes
`min`	number	Yes	Zone start value
`max`	number	Yes	Zone end value
`color`	string	Yes	Hex color
`label`	string	No	Optional label

Example: Create Thermostat Gauge (Dual Setpoints + Current Pointer + Humidity)

```
{
  "command": "create_gauge",
  "gauge_id": "upstairs-thermostat",
  "title": "Upstairs",
  "gauge_type": "thermostat",
  "min": 50,
  "max": 90,
  "unit": "°F",
  "step_size": 1,
  "mqtt_topic_prefix": "kingkiosk",
  "pointers": [
    {
      "id": "current",
      "label": "Current",
      "style": "needle",
      "color": "#E5E7EB",
      "locked": true,
      "subscribe_topic": "kingkiosk/ha/state/climate/upstairs",
      "json_field": "attributes.current_temperature"
    },
    {
      "id": "heat",
      "label": "Heat",
      "style": "dot",
      "color": "#FF6B35",
      "locked": false,
      "subscribe_topic": "kingkiosk/ha/state/climate/upstairs",
      "publish_topic": "kingkiosk/ha/command/climate/upstairs",
      "json_field": "attributes.target_temp_low"
    },
    {
      "id": "cool",
      "label": "Cool",
      "style": "dot",
      "color": "#4FC3F7",
      "locked": false,
      "subscribe_topic": "kingkiosk/ha/state/climate/upstairs",
      "publish_topic": "kingkiosk/ha/command/climate/upstairs",
      "json_field": "attributes.target_temp_high"
    },
    {
      "id": "humidity",
      "label": "Humidity",
      "style": "dot",
      "color": "#94A3B8",
      "locked": true,
      "subscribe_topic": "kingkiosk/ha/state/climate/upstairs",
      "json_field": "attributes.current_humidity"
    }
  ]
}
```

Thermostat dual-mode detection: - `current < heat` => `Heating` - `current > cool` => `Cooling` - otherwise => `Idle`

Thermostat center text behavior: - Dual setpoint: `HEAT ... COOL`, `DRAG DOT TO ADJUST`, plus mode text (`Heating`/`Cooling`/`Idle`) - If a humidity pointer is present, the center also shows `HUMIDITY xx%`

Example: Create Gauge with Native OS Widget

```
{
  "command": "create_gauge",
  "gauge_id": "outdoor_temp",
  "title": "Outdoor Temperature",
  "gauge_type": "radial",
  "min": -20,
  "max": 120,
  "unit": "°F",
  "decimals": 1,
  "color_mode": "thresholds",
  "thresholds": [
    {"value": 32, "color": "#3498db", "label": "Freezing"},
    {"value": 70, "color": "#2ecc71", "label": "Comfortable"},
    {"value": 90, "color": "#e74c3c", "label": "Hot"}
  ],
  "os_widget": true,
  "mqtt_topic": "weather/outdoor/temperature",
  "json_field": "temp_f"
}
```

This creates both an in-app gauge and registers it as a native OS widget that can be added to the Android home screen or iOS home/lock screen. The widget independently subscribes to `weather/outdoor/temperature` and extracts the value from the `temp_f` JSON field.

0.1.13.12.4 Update Gauge Value (`set_value`, `set_gauge_value`, `update_gauge_value`)

Key	Type	Required	Notes
`gauge_id`	string	Yes*	Gauge identifier
`window_id`	string	Yes*	Alias for `gauge_id`
`value`	number	Yes	New value to display
`mqtt_topic_prefix`	string	No	Default `"kingkiosk"`. Used to construct controller tag.

*Either `gauge_id` or `window_id` is required

0.1.13.12.5 Configure Gauge (`set_gauge_config`, `configure_gauge`)

Key	Type	Required	Notes
`gauge_id` / `window_id`	string	Yes*	Gauge identifier
`config`	object	No	Optional nested config object (merged with top-level keys).

Key	Type	Required	Notes
`mqtt_topic_prefix`	string	No	Default `"kingkiosk"`. Used to construct controller tag.
`min`	number	No	Minimum value
`max`	number	No	Maximum value
`value`	number	No	Current value
`unit`	string	No	Unit label
`label`	string	No	Additional label text
`gauge_type`	string	No	Display style
`interactive`	bool	No	Allow user interaction
`locked`	bool	No	Lock value display
`step_size`	number	No	Step increment
`decimals`	number	No	Decimal places
`show_min_max`	bool	No	Show min/max labels
`show_value`	bool	No	Show current value
`color_mode`	string	No	`"gradient"`, `"thresholds"`, `"solid"`, `"zones"`
`thresholds`	array	No	Color threshold definitions
`zones`	array	No	Color zone definitions
`pointers`	array	No	Pointer definitions

0.1.13.12.6 Lock/Unlock Commands **`lock_gauge`**: Lock gauge to prevent user interaction

```
{
  "command": "lock_gauge",
  "gauge_id": "thermostat-1"
}
```

`unlock_gauge`: Unlock gauge to allow user interaction

`toggle_gauge_lock`: Toggle the lock state

0.1.13.12.7 Multi-Pointer Support

For thermostat-style gauges with multiple indicators (current temperature + setpoints):

Pointer Properties:

Property	Type	Required	Default	Notes
`id`	string	Yes	-	Unique pointer identifier
`value`	number	No	`0`	Initial value
`label`	string	No	`""`	Pointer label
`color`	string	No	`"#00BFFF"`	Hex color
`icon`	string	No	-	Icon name
`locked`	bool	No	`false`	Prevent user adjustment
`style`	string	No	`"needle"`	`"needle"`, `"dot"`, `"triangle"`, `"line"`, `"target"`
`subscribe_topic`	string	No	-	MQTT topic to receive value updates

Property	Type	Required	Default	Notes
`publish_topic`	string	No	-	MQTT topic to publish user changes
`publish_payload`	object	No	-	Optional custom payload template. Supports `{{value}}`, `{{pointer_id}}`, `{{gauge_id}}`, `{{timestamp}}`, `{{current}}`, `{{heat}}`, `{{cool}}` tokens.
`json_field`	string	No	-	Dot-path field extractor for JSON payloads (aliases: `json_path`, `value_path`)

For Home Assistant climate payloads, a common mapping is: - `current` -> `attributes.current_temperatur`
- `heat/low` -> `attributes.target_temp_low` - `cool/high` -> `attributes.target_temp_high`
- `humidity -> attributes.current_humidity` (renders as center text on thermostat gauges)

Auxiliary pointer values such as humidity are not clamped to gauge `min`/`max`; temperature-related pointers continue to clamp to the configured range.

All pointers may share one `subscribe_topic`; the controller de-duplicates subscriptions and updates each pointer using its own `json_field`.

Interaction behavior for thermostat multi-pointer gauges: - Drag must begin on/near an unlocked setpoint handle to adjust - Tap/click on the dial background does not change setpoints

When `publish_topic` matches `*/ha/command/climate/*` and `publish_payload` is not provided, the gauge now publishes Home Assistant bridge-compatible commands by default:

```
{
  "service": "set_temperature",
  "data": {
    "target_temp_low": 68,
    "target_temp_high": 76
  }
}
```

For single-setpoint pointers, fallback payload is:

```
{
  "service": "set_temperature",
  "data": {
    "temperature": 72
  }
}
```

Custom range thermostat `publish_payload` examples:

Heat pointer template:

```
{
  "id": "heat",
  "publish_topic": "kingkiosk/ha/command/climate/upstairs",
  "publish_payload": {
    "service": "set_temperature",
    "data": {
      "target_temp_low": "{{value}}",
      "target_temp_high": "{{cool}}"
    }
  }
}
```

Cool pointer template:

```
{
  "id": "cool",
  "publish_topic": "kingkiosk/ha/command/climate/upstairs",
  "publish_payload": {
    "service": "set_temperature",
    "data": {
      "target_temp_low": "{{heat}}",
      "target_temp_high": "{{value}}"
    }
  }
}
```

Template variables accepted in `publish_payload`: - `{{value}}`, `{{pointer_id}}`, `{{gauge_id}}`, `{{timestamp}}` - `{{current}}`, `{{heat}}`, `{{cool}}`

`set_pointer_value`: Update a specific pointer's value

```
{
  "command": "set_pointer_value",
  "gauge_id": "nest-thermostat",
  "pointer_id": "current",
  "value": 72
}
```

`add_pointer`: Add a new pointer to a gauge

```
{
  "command": "add_pointer",
  "gauge_id": "nest-thermostat",
  "pointer": {
    "id": "humidity",
    "label": "Humidity",
    "color": "#9b59b6",
    "locked": true,
    "style": "dot"
  }
}
```

`remove_pointer`: Remove a pointer from a gauge

```
{
  "command": "remove_pointer",
  "gauge_id": "nest-thermostat",
  "pointer_id": "humidity"
}
```

0.1.13.12.8 Other Gauge Commands

- `delete_gauge`: Remove a gauge widget
- `list_gauges`: List all active gauge instances

0.1.13.12.9 State Publishing When `mqtt_topic_prefix` is set, the gauge publishes state updates to:

`{mqtt_topic_prefix}/state`

Published State Format:

```
{
  "widget_id": "thermostat-1",
  "type": "gauge",
  "value": 72,
  "min": 50,
  "max": 90,
  "percentage": 55,
  "unit": "°F",
  "label": "Living Room",
  "style": "thermostat",
  "gauge_type": "thermostat",
  "color_mode": "zones",
  "decimals": 0,
  "formatted_value": "72 °F",
  "interactive": true,
  "locked": false,
  "step_size": 1,
  "show_min_max": true,
  "show_value": true,
  "pointers": [
    { "id": "current", "value": 72, "locked": true },
    { "id": "setpoint_high", "value": 76, "locked": false }
  ],
  "current_threshold": {
    "value": 70,
    "color": "#2ecc71",
    "label": "Comfort"
  },
  "timestamp": 1704153600
}
```

0.1.13.12.10 User Interaction Publishing When a user adjusts an interactive pointer, the gauge publishes to:

`{mqtt_topic_prefix}/user_input`

```
{
  "gauge_id": "nest-thermostat",
  "pointer_id": "setpoint_high",
  "value": 75,
  "previous_value": 76,
  "timestamp": 1704153600
}
```

0.1.13.12.11 MQTT Topics

Topic	Direction	Notes
`{prefix}/gauge/{gaugeId}/value`	Subscribe	Receive value updates
`{prefix}/gauge/{gaugeId}/config`	Subscribe	Receive configuration
`{prefix}/gauge/{gaugeId}/status`	Publish	Publish status updates
`{prefix}/gauge/{gaugeId}/set`	Publish	Publish user-set values
`{prefix}/state`	Publish	Full state (retained)
`{prefix}/user_input`	Publish	User interaction events

0.1.13.12.12 Platform Notes

- **tvOS**: Use Siri Remote D-pad to select pointers and adjust values
- **iOS**: Touch/swipe on gauge to adjust, tap pointers to select
- **macOS**: Keyboard arrows to adjust, mouse click to select pointers

---

0.1.13.13 Carousels

0.1.13.13.1 Create (`create_carousel`, `create_video_carousel`, `create_image_carousel`, `create_widget_carousel`)

Key	Type	Notes
`window_id`	string	Required.
`title`	string	Optional.
`items`	array	Optional.
`config`	object	Optional; see below.
`opacity`, `x`, `y`, `width`, `height`	number	Optional geometry.

Carousel `config` keys:

Key	Type	Notes
auto_play	bool	-
interval	int	-
viewport_fraction	number	-
infinite_scroll	bool	-
reverse	bool	-
scroll_direction	string	vertical or horizontal.
enlarge_center_page	bool	-
show_indicator	bool	-
pause_on_interaction	bool	-
resume_timeout	int	-
enable_manual_control	bool	-
disable_center	bool	-
pad_ends	bool	-
page_snapping	bool	-
layout_mode	string	-

Other carousel commands: add_carousel_item, remove_carousel_item, update_carousel, delete_carousel, list_carousels, plus navigation (navigate_carousel/goto_carousel) with index or target_id and optional animate + duration (ms).

Config/status commands: set_carousel_config, get_carousel_status.

- Both route to the same implementation which **updates** the carousel configuration by window_id.
- get_carousel_status currently does not publish a status payload; it simply returns a generic {success:true, command, timestamp} on response_topic.

Top-level keys for set_carousel_config / get_carousel_status:

Key	Type	Notes
window_id	string	Required.

Key	Type	Notes
`auto_play`	bool	Optional.
`interval`	int	Optional.
`viewport_fraction`	number	Optional.
`infinite_scroll`	bool	Optional.
`reverse`	bool	Optional.
`scroll_direction`	string	`vertical` or `horizontal`. Note: omitting this defaults to `horizontal` (overrides any existing value).
`enlarge_center_page`	bool	Optional.
`show_indicator`	bool	Optional.
`pause_on_interaction`	bool	Optional.
`resume_timeout`	int	Optional.
`enable_manual_control`	bool	Optional.

0.1.13.14 Media (Video/Audio/Image/Web)

0.1.13.14.1 `play_media`

Key	Type	Default	Notes
`type`	string	inferred	`video`, `audio`, `image`, `web`, `webrtc`.
`url`	string	(required)	Media URL.

Key	Type	Default	Notes
`style`	string	-	For audio: `window` or `visualizer`; for video: `window` or `fullscreen` (fullscreen not implemented).
`loop`	bool/string	`false`	Truthy string accepted.
`window_id`	string	-	For tiled display.
`title`	string	varies	Defaults: `Kiosk Video` / `Kiosk Audio` / `MQTT Image` / `WebRTC Stream`.
`opacity`, `x`, `y`, `width`, `height`	number	-	Optional geometry.
`hardware_accel`	bool/string	-	Temporary hardware accel preference for this request.
`allow_bad_cert`	bool/string	`false`	Applies to background audio playback (`type: audio` without window style).

Audio visualizer options (when `type == audio` and `style == visualizer`):

Key	Type	Default
`visualizer_type`	string	`fft`
`bars`	int	`64`
`smoothing`	number	`0.8`

Key	Type	Default
`color_scheme`	string	`rainbow`
`show_peaks`	bool	`true`
`peak_decay`	number	`0.95`
`update_frequency`	int	`60`

0.1.13.14.2 `youtube`

Key	Type	Default	Notes
`url`	string	(required)	YouTube URL.
`title`	string	`YouTube`	-
`window_id`	string	(auto)	If omitted, auto-generated.
`opacity`, `x`, `y`, `width`, `height`	number	-	Optional geometry.

0.1.13.14.3 Media window control (`play`, `pause`, `close`, plus `enter_fullscreen`/`exit_fullscreen`/`tog`

These are window-id based and routed to the appropriate window controller.

Legacy compatibility:

- `pause_media` is accepted but **deprecated**. It logs a warning and routes to the same handler as `{command:'pause', window_id: ...}`.

0.1.13.14.4 Background audio control (`play_audio`, `pause_audio`, `stop_audio`, `seek_audio`)

These commands control the background audio playback service.

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	One of `play_audio`, `pause_audio`, `stop_audio`, `seek_audio`.
`url`	string	-	For `play_audio`: optional URL to play. If omitted, resumes current audio.
`loop`	bool/string	`false`	For `play_audio`: loop playback. Truthy string accepted.
`allow_bad_cert`	bool/string	`false`	For `play_audio`: allow invalid SSL certificates.
`position`	number/string	`0`	Only used by `seek_audio` (seconds).

Notes:

- These handlers perform actual media control actions via `MediaControlService`, but do not publish MQTT success/error responses.

0.1.13.14.5 Emergency media reset (`reset_media`) This triggers the media recovery service to reset media resources.

Top-level keys:

Key	Type	Default	Notes
`command`	string	(required)	Must be `reset_media`.
`force`	bool/string	`false`	If true, forces reset behavior in the recovery service.
`test`	bool/string	`false`	If true, runs a health report only (no reset).

Published topics:

- If `test == true`, publishes health status to `kingkiosk/{device}/status/media_health`.
- If a reset is performed successfully, publishes a report to `kingkiosk/{device}/status/media_reset` with: `{success:true, timestamp, resetCount, forced, audioRestored, audioUrl}`.

Notes:

- The handler attempts to capture/restore background audio across the reset when possible.

0.1.13.15 Web / PDF

0.1.13.15.1 `open_browser` / `open_web` / `open_simple_web`

Key	Type	Default
`url` / `initial_url`	string	(required)
`title`	string	`Simple Web`
`window_id`	string	(auto)
`opacity`, `x`, `y`, `width`, `height`	number	-

Note (tvOS): Apple TV maps `open_browser` / `open_web` / `open_simple_web` to `create_remote_browser`.

Note (Flutter): open_browser / open_web are aliases for open_simple_web (SimpleWeb).

0.1.13.15.2 **webrtc_player**

Opens a native WHEP WebRTC stream tile for low-latency streaming.

Key	Type	Default
url	string	(required)
webrtc_url	string	-
title	string	WebRTC Player
window_id	string	(auto)
opacity, x, y, width, height	number	-

Example:

```json
{
  "command": "webrtc_player",
  "url": "http://192.168.0.199:1984/webrtc.html?src=Backyard_Camera",
  "title": "Backyard Camera",
  "x": 100,
  "y": 100,
  "width": 640,
  "height": 480
}
```

0.1.13.15.3 **open_pdf**

Key	Type	Default
url	string	(required)
title	string	PDF Document
window_id	string	(auto)
opacity, x, y, width, height	number	-

Note: runtime web/PDF actions (refresh, paging, etc.) are not currently exposed as a canonical MQTT command surface. Treat these windows as configured via their open_* system commands.

0.1.13.16 Calendar

System command: `calendar`

Top-level keys:

Key	Type	Default	Notes
`action`	string	-	`show`, `create`, `hide`, `add_event`, `remove_event`, `clear_events`, `go_to_date`, `format`.
`name`	string	`Calendar`	-
`window_id`	string	(auto)	Used for create/hide.
`opacity`, `x`, `y`, `width`, `height`	number	-	Optional geometry.

Event management actions are forwarded to `CalendarController.handleMqttCalendarCommand(...).`

0.1.13.17 Timers / Stopwatch

System commands:

Command	Keys
`stopwatch`	`name`, `window_id`, `config` (object), plus common geometry keys
`timer_widget`	`name`, `window_id`, `config` (object), plus common geometry keys
`timer_control`	`timer_id` (required), `action` (required), plus any additional keys forwarded to the timer window

---

0.1.13.18 Games

System commands:

Command	Keys
`stop_the_missiles`	`title`, optional `window_id`, optional `game_type` (default `missile_command`), optional `config` (object), optional `opacity`. Note: geometry keys (`x`, `y`, `width`, `height`) are parsed but **not passed** to the tile creator.
`game_control`	`window_id`, `action` (`start/restart/stop/pause/resume/toggle_sou` plus payload forwarded
`close_all_games`	No keys. Closes all game tiles.
`game_state_query`	`window_id` (required). Publishes game state to `kingkiosk/{device}/game_state`.

0.1.13.19 MQTT Image Tile

System command: `mqtt_image`

This command creates and manages a tile that updates its displayed image based on MQTT messages.

Top-level keys used:

Key	Type	Default	Notes
`command`	string	(required)	Must be `mqtt_image`.

Key	Type	Default	Notes
`action`	string	`open`	`open/create`, `update_topic`, `close`.
`window_name`	string	auto	Preferred name key.
`name`	string	-	Alias for `window_name`.
`window_id`	string	-	If provided, used as the tile ID and registers a window controller for basic window actions. Otherwise, `window_name` is used as the tile ID.
`mqtt_topic`	string	(required)	Topic to subscribe for image updates.
`json_field`	string	-	Optional. Extracts image data from JSON using dot notation (e.g. `data.image`). If omitted, common keys are auto-detected.
`is_base64`	bool/string	`false`	Parsed by `toString().toLowerCase() == 'true'`.
`initial_image /url`	string	-	Optional initial display content.
`update_interval`	int/string	`0`	Milliseconds; stored on the tile when > `0`.
`opacity`	number/string	`1.0`	-

Key	Type	Default	Notes
x, y	number/string	100	-
width, height	number/string	800 / 600	-
response_topic	string	kingkiosk/{device}/system/response	Receives response {success, command:'mqtt_image', action, window_name, timestamp}.

Image payload behavior:

- If json_field is provided (or payload looks like JSON), the handler tries to parse JSON and extract the image value.
- If is_base64 == true and the extracted value is not a data: URL, the handler will prefix data:image/png;base64,.
- If is_base64 == false, the handler may still auto-detect large base64 payloads and treat them as data:image/png;base64,....

0.1.14 Integration Examples

0.1.14.1 Node-RED: Monitor and Control a Clock Widget

```
[
    {
        "id": "clock-state-sub",
        "type": "mqtt in",
    "topic": "kingkiosk/my-device/element/clock-1/state",
        "qos": "1"
    },
    {
        "id": "clock-cmd-pub",
        "type": "mqtt out",
    "topic": "kingkiosk/my-device/element/clock-1/cmd",
        "qos": "1"
    }
]
```

0.1.14.2 Home Assistant: Widget State Sensor

```
mqtt:
  sensor:
    - name: "Living Room Clock Mode"
      state_topic: "kingkiosk/my-device/element/clock-1/state"
      value_template: "{{ value_json.mode }}"
      json_attributes_topic: "kingkiosk/my-device/element/clock-1/state"

  button:
    - name: "Toggle Clock Mode"
      command_topic: "kingkiosk/my-device/element/clock-1/cmd"
      payload_press: '{"command": "toggle_mode"}'
```

0.1.14.3 Python: List All Widgets on a Device

```
import paho.mqtt.client as mqtt
import json

def on_message(client, userdata, msg):
    info = json.loads(msg.payload)
    print(f"Device: {info['device_id']}")
    print(f"Active widgets: {info['active_widgets']}")
    for widget_id in info['active_widgets']:
        print(f"  - {widget_id}")

client = mqtt.Client()
client.on_message = on_message
client.connect("broker.local", 1883)
client.subscribe("kingkiosk/+/info")
client.loop_forever()
```

0.1.15 Canonical Topic Summary

- Send device-wide commands to `kingkiosk/{device_id}/system/cmd`
- Send element-scoped commands (when supported) to `kingkiosk/{device_id}/element/{element_`
- Subscribe for responses on `kingkiosk/{device_id}/system/response` and/or `kingkiosk/{device_id}/element/{element_id}/response`
- Subscribe for retained element state on `kingkiosk/{device_id}/element/{element_id}/state`

0.1.16 Developer Guide: Adding MQTT Support to Widgets

To add per-widget MQTT support to a widget controller:

0.1.16.1 1. Add the Mixin

```
import '../../../widgets/mqtt_widget_mixin.dart';

class MyWidgetController extends GetxController
    with MqttWidgetMixin
    implements KioskWindowController {

  @override
  String get widgetId => windowName;

  @override
  String get widgetType => 'my_widget';
```

0.1.16.2 2. Register in onInit

```
@override
void onInit() {
  super.onInit();
  registerWithMqtt();
}
```

0.1.16.3 3. Unregister in onClose

```
@override
void onClose() {
  unregisterFromMqtt();
  super.onClose();
}
```

0.1.16.4 4. Handle Commands

```
@override
Future<Map<String, dynamic>?> handleMqttCommand(
    Map<String, dynamic> command) async {
```

```
  final cmd = command['command'] as String?;

  switch (cmd) {
    case 'my_command':
      doSomething();
      return {'status': 'success'};
    default:
      return null; // Let mixin handle common commands
  }
}
```

0.1.16.5 5. Build State

```
@override
Map<String, dynamic> buildState() {
  return {
    'type': widgetType,
    'widget_id': widgetId,
    'my_value': myValue,
    // ... other state fields
  };
}
```

0.1.16.6 6. Publish Events (Optional)

```
// Publish custom events
publishEvent({'event': 'my_event', 'data': someData});

// Convenience methods
publishSimpleEvent('clicked');
publishError('Something went wrong', 'ERR_CODE');
publishStateChange('idle', 'playing');
```