

```
//@version=6
```

```
indicator("MFB - Levels Coherence", overlay=true, max_boxes_count=500,  
max_lines_count=500, max_labels_count=500)
```

```
// --- Inputs ---
```

```
group_sessions = "Sessions (EST/EDT)"
```

```
en_ny = input.bool(true, "NY Open", group=group_sessions, inline="ny")
```

```
ny_sess_time = input.session("0930-1100", "", group=group_sessions, inline="ny")
```

```
en_lon = input.bool(true, "London Open", group=group_sessions, inline="lon")
```

```
lon_sess_time = input.session("0300-0430", "", group=group_sessions, inline="lon")
```

```
en_tok = input.bool(true, "Tokyo Open", group=group_sessions, inline="tok")
```

```
tok_sess_time = input.session("1900-2030", "", group=group_sessions, inline="tok")
```

```
en_nya = input.bool(true, "NY Afternoon", group=group_sessions, inline="nya")
```

```
nya_sess_time = input.session("1500-1630", "", group=group_sessions, inline="nya")
```

```
en_comex = input.bool(false, "Comex (Gold)", group=group_sessions, inline="comex")
```

```
comex_sess_time= input.session("0820-0850", "", group=group_sessions, inline="comex")
```

```
asia_sess_time = input.session("1800-0000", "Asia Session", group=group_sessions)
```

```
lon_full_time = input.session("0000-0600", "London Full Session", group=group_sessions)
```

```
group_logic = "Strategy Logic"
```

```
points_dist = input.float(12.0, "Confluence Distance (Points)", group=group_logic)
```

```

group_display = "Display Settings"

show_info_table = input.bool(false, "Show Discernment Table", group=group_display)

show_aoi_score = input.bool(false, "Show AOI Score Label", group=group_display)

show_15m_or = input.bool(true, "Show 15m OR Lines", group=group_display)

or_15_width = input.int(1, "15m OR Line Width", minval=1, maxval=10,
group=group_display)

show_30m_or = input.bool(true, "Show 30m OR Lines", group=group_display)

or_30_width = input.int(1, "30m OR Line Width", minval=1, maxval=10,
group=group_display)

show_1h_or = input.bool(true, "Show 1h OR Lines", group=group_display)

or_1h_width = input.int(1, "1h OR Line Width", minval=1, maxval=10,
group=group_display)

show_pathway = input.bool(false, "Show Pathway Arrows", group=group_display,
tooltip="Displays continuation arrows when price accepts outside the OR and is far from
external liquidity.")

// --- Timezones & Sessions ---

tz = "America/New_York"

in_session(sess) => not na(time(timeframe.period, sess, tz))

is_new_session(sess) => in_session(sess) and not in_session(sess)[1]

ny_start = en_ny and is_new_session(ny_sess_time)

lon_start = en_lon and is_new_session(lon_sess_time)

tok_start = en_tok and is_new_session(tok_sess_time)

nya_start = en_nya and is_new_session(nya_sess_time)

comex_start = en_comex and is_new_session(comex_sess_time)

```

```
any_start = ny_start or lon_start or tok_start or nya_start or comex_start
```

```
// --- Session State & OR ---
```

```
var float sess_start_time = na
```

```
var float or_15_h = na, var float or_15_l = na
```

```
var float or_30_h = na, var float or_30_l = na
```

```
var float or_1h_h = na, var float or_1h_l = na
```

```
if any_start
```

```
    sess_start_time := time
```

```
    or_15_h := high, or_15_l := low
```

```
    or_30_h := high, or_30_l := low
```

```
    or_1h_h := high, or_1h_l := low
```

```
// Update ORs based on elapsed time from session start
```

```
time_elapsed_mins = (time - sess_start_time) / 60000
```

```
if time_elapsed_mins < 15
```

```
    or_15_h := math.max(or_15_h, high)
```

```
    or_15_l := math.min(or_15_l, low)
```

```
if time_elapsed_mins < 30
```

```
    or_30_h := math.max(or_30_h, high)
```

```
    or_30_l := math.min(or_30_l, low)
```

```
if time_elapsed_mins < 60
```

```
    or_1h_h := math.max(or_1h_h, high)
```

```
or_1h_l := math.min(or_1h_l, low)

// Extend NY OR until 12:00 PM ET
in_ny_extension = not na(time(timeframe.period, "0930-1200", tz))
in_comex_extension = not na(time(timeframe.period, "0820-1200", tz))

// Create variables to track if a specific session's lines should be drawing
var bool ny_lines_active = false
var bool lon_lines_active = false
var bool tok_lines_active = false
var bool nya_lines_active = false
var bool comex_lines_active = false

if ny_start
    ny_lines_active := true
    lon_lines_active := false
    tok_lines_active := false
    nya_lines_active := false
    comex_lines_active := false
else if lon_start
    lon_lines_active := true
    ny_lines_active := false
    tok_lines_active := false
    nya_lines_active := false
    comex_lines_active := false
else if tok_start
```

```
tok_lines_active := true
ny_lines_active := false
lon_lines_active := false
nya_lines_active := false
comex_lines_active := false
else if nya_start
  nya_lines_active := true
  ny_lines_active := false
  lon_lines_active := false
  tok_lines_active := false
  comex_lines_active := false
else if comex_start
  comex_lines_active := true
  ny_lines_active := false
  lon_lines_active := false
  tok_lines_active := false
  nya_lines_active := false

// Check if we are past the end of the current session's extension
if ny_lines_active and not in_ny_extension
  ny_lines_active := false
if lon_lines_active and not in_session(lon_sess_time)
  lon_lines_active := false
if tok_lines_active and not in_session(tok_sess_time)
  tok_lines_active := false
if nya_lines_active and not in_session(nya_sess_time)
```

```
    nya_lines_active := false
if comex_lines_active and not in_comex_extension
    comex_lines_active := false

can_extend_lines = ny_lines_active or lon_lines_active or tok_lines_active or
nya_lines_active or comex_lines_active

// Determine if signals are allowed (Option A: only during active sessions)
signals_allowed = can_extend_lines

// Visualizing the ORs
var line or_15_h_line = na
var line or_15_l_line = na
var line or_30_h_line = na
var line or_30_l_line = na
var line or_1h_h_line = na
var line or_1h_l_line = na

if any_start
    or_15_h_line := na
    or_15_l_line := na
    or_30_h_line := na
    or_30_l_line := na
    or_1h_h_line := na
    or_1h_l_line := na
```

if show_15m_or

if time_elapsed_mins >= 15 and not na(or_15_h) and na(or_15_h_line)

or_15_h_line := line.new(bar_index - int(time_elapsed_mins), or_15_h, bar_index, or_15_h, color=color.red, style=line.style_solid, width=or_15_width)

or_15_l_line := line.new(bar_index - int(time_elapsed_mins), or_15_l, bar_index, or_15_l, color=color.red, style=line.style_solid, width=or_15_width)

else if time_elapsed_mins >= 15 and can_extend_lines and not na(or_15_h_line)

line.set_x2(or_15_h_line, bar_index)

line.set_x2(or_15_l_line, bar_index)

if show_30m_or

if time_elapsed_mins >= 30 and not na(or_30_h) and na(or_30_h_line)

or_30_h_line := line.new(bar_index - int(time_elapsed_mins), or_30_h, bar_index, or_30_h, color=color.red, style=line.style_solid, width=or_30_width)

or_30_l_line := line.new(bar_index - int(time_elapsed_mins), or_30_l, bar_index, or_30_l, color=color.red, style=line.style_solid, width=or_30_width)

else if time_elapsed_mins >= 30 and can_extend_lines and not na(or_30_h_line)

line.set_x2(or_30_h_line, bar_index)

line.set_x2(or_30_l_line, bar_index)

if show_1h_or

if time_elapsed_mins >= 60 and not na(or_1h_h) and na(or_1h_h_line)

or_1h_h_line := line.new(bar_index - int(time_elapsed_mins), or_1h_h, bar_index, or_1h_h, color=color.red, style=line.style_solid, width=or_1h_width)

or_1h_l_line := line.new(bar_index - int(time_elapsed_mins), or_1h_l, bar_index, or_1h_l, color=color.red, style=line.style_solid, width=or_1h_width)

else if time_elapsed_mins >= 60 and can_extend_lines and not na(or_1h_h_line)

line.set_x2(or_1h_h_line, bar_index)

```
line.set_x2(or_1h_l_line, bar_index)

// --- Key Levels (PDH, PDL, PWH, PWL, PMH, PML, Asia, London) ---
[pdh, pdl] = request.security(syminfo.tickerid, "D", [high[1], low[1]],
lookahead=barmerge.lookahead_on)

[pwh, pwl] = request.security(syminfo.tickerid, "W", [high[1], low[1]],
lookahead=barmerge.lookahead_on)

[pmh, pml] = request.security(syminfo.tickerid, "M", [high[1], low[1]],
lookahead=barmerge.lookahead_on)

var float asia_h = na, var float asia_l = na
var float lon_h = na, var float lon_l = na

if is_new_session(asia_sess_time)
    asia_h := high
    asia_l := low
else if in_session(asia_sess_time)
    asia_h := math.max(asia_h, high)
    asia_l := math.min(asia_l, low)

if is_new_session(lon_full_time)
    lon_h := high
    lon_l := low
else if in_session(lon_full_time)
    lon_h := math.max(lon_h, high)
    lon_l := math.min(lon_l, low)
```

```

// Function to calculate confluence
get_confluence(float price) =>
    float nearest = na
    float min_dist = 999999.0
    int score = 0

    levels = array.from(pdh, pdl, pwh, pwl, pmh, pml, asia_h, asia_l, lon_h, lon_l)
    for lvl in levels
        if not na(lvl)
            dist = math.abs(price - lvl)
            if dist < min_dist
                min_dist := dist
                nearest := lvl
            if dist <= points_dist
                score += 1
    [nearest, min_dist, score]

[nearest_to_or_h, dist_h, score_h] = get_confluence(or_15_h)
[nearest_to_or_l, dist_l, score_l] = get_confluence(or_15_l)

is_aoi_h = dist_h <= points_dist
is_aoi_l = dist_l <= points_dist

// --- FVG-123 Logic ---
raw_fvg_bull = low > high[2] and close[1] > open[1]
raw_fvg_bear = high < low[2] and close[1] < open[1]

```

```
var float active_bull_fvg_top = na
var float active_bull_fvg_bot = na
var bool bull_retrace = false
```

```
var float active_bear_fvg_bot = na
var float active_bear_fvg_top = na
var bool bear_retrace = false
```

```
if raw_fvg_bull
  active_bull_fvg_top := low
  active_bull_fvg_bot := high[2]
  bull_retrace := false
```

```
if raw_fvg_bear
  active_bear_fvg_bot := high
  active_bear_fvg_top := low[2]
  bear_retrace := false
```

```
if not na(active_bull_fvg_top) and low <= active_bull_fvg_top
  bull_retrace := true
```

```
if not na(active_bear_fvg_bot) and high >= active_bear_fvg_bot
  bear_retrace := true
```

```
fvg123_bull = false
```

```
fvg123_bear = false
```

```
if bull_retrace and close > active_bull_fvg_top
```

```
    fvg123_bull := true
```

```
    active_bull_fvg_top := na
```

```
if bear_retrace and close < active_bear_fvg_bot
```

```
    fvg123_bear := true
```

```
    active_bear_fvg_bot := na
```

```
// --- Sweep/Reversal Logic Implementation ---
```

```
var bool swept_h = false
```

```
var float sweep_h_val = na
```

```
var bool swept_l = false
```

```
var float sweep_l_val = na
```

```
if is_aoi_h and high > math.max(or_15_h, nearest_to_or_h)
```

```
    swept_h := true
```

```
    sweep_h_val := na(sweep_h_val) ? high : math.max(sweep_h_val, high)
```

```
if is_aoi_l and low < math.min(or_15_l, nearest_to_or_l)
```

```
    swept_l := true
```

```
    sweep_l_val := na(sweep_l_val) ? low : math.min(sweep_l_val, low)
```

```
long_sweep_setup = swept_l and close > or_15_l and fvg123_bull and signals_allowed
```

```
short_sweep_setup = swept_h and close < or_15_h and fvg123_bear and signals_allowed
```

```

if long_sweep_setup
    swept_l := false
    sweep_l_val := na
if short_sweep_setup
    swept_h := false
    sweep_h_val := na

// --- Pathway Logic (Continuation) ---
is_pathway_h = dist_h > points_dist
is_pathway_l = dist_l > points_dist

long_pathway_setup = is_pathway_h and close > or_15_h and fvg123_bull and
signals_allowed
short_pathway_setup = is_pathway_l and close < or_15_l and fvg123_bear and
signals_allowed

// --- Visualizing Entries & Lines ---
plotshape(long_sweep_setup, "Long Sweep Entry", shape.triangleup, location.belowbar,
color.green, size=size.small)

plotshape(short_sweep_setup, "Short Sweep Entry", shape.triangledown,
location.abovebar, color.red, size=size.small)

plotshape(show_pathway ? long_pathway_setup : false, "Long Pathway Entry",
shape.arrowup, location.belowbar, color.lime, size=size.small)

plotshape(show_pathway ? short_pathway_setup : false, "Short Pathway Entry",
shape.arrowdown, location.abovebar, color.maroon, size=size.small)

// Move AOI labels to the right

```

```

if long_sweep_setup and score_l > 0 and show_aoi_score

    label.new(bar_index + 1, low, "AOI Score: " + str.tostring(score_l), color=#00000000,
textcolor=color.green, style=label.style_label_left)

if short_sweep_setup and score_h > 0 and show_aoi_score

    label.new(bar_index + 1, high, "AOI Score: " + str.tostring(score_h), color=#00000000,
textcolor=color.red, style=label.style_label_left)

// --- Information Table ---

var table infoTable = table.new(position.bottom_center, 2, 3,
bgcolor=color.new(color.black, 70), border_color=color.gray, border_width=1)

if barstate.islast and show_info_table

    table.cell(infoTable, 0, 0, "Trade Discernment", text_color=color.white,
text_size=size.small, text_halign=text.align_center, bgcolor=color.new(color.blue, 70))

    table.merge_cells(infoTable, 0, 0, 1, 0)

    table.cell(infoTable, 0, 1, "Confluence AOI", text_color=color.green, text_size=size.small)

    table.cell(infoTable, 1, 1, "OR & Ext Liq ≤ 12 pts (Reversal/Sweep)", text_color=color.white,
text_size=size.small, text_halign=text.align_left)

    table.cell(infoTable, 0, 2, "Pathway Mode", text_color=color.lime, text_size=size.small)

    table.cell(infoTable, 1, 2, "OR & Ext Liq > 12 pts (Continuation)", text_color=color.white,
text_size=size.small, text_halign=text.align_left)

// --- Alerts ---

alert_coherence = long_sweep_setup or short_sweep_setup

alertcondition(alert_coherence, title="MFB - Coherence", message="MFB Coherence
Sweep setup triggered on {{ticker}}")

```