

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

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

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

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

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

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

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

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

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

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

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

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

```
en_comex    = input.bool(true, "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)
```

```
rrRatio       = input.float(2.0, "Risk/Reward Ratio", step = 0.5,  
group=group_logic)
```

```
group_signals = "Signal Toggles"
```

```
trading_mode  = input.string("Custom", "Trading Mode Preset",  
options=["Custom", "1 min 9 EMA with HTF Support", "5 min Signals Only"],  
group=group_signals, tooltip="Select a preset to automatically configure  
signals, or select 'Custom' to use the checkboxes below.")
```

```
only_ema      = input.bool(false, "Show ONLY 9 EMA Retrace Signals",  
group=group_signals, tooltip="Overrides all other options to only show 9 EMA  
Retrace signals.")
```

```
only_cisd     = input.bool(false, "Show ONLY CISC Signals",  
group=group_signals, tooltip="Overrides all other options to only show 5m  
CISC signals.")
```

```
show_sweep = input.bool(true, "Enable Sweep Signals (Custom)",
group=group_signals)

show_pathway = input.bool(true, "Enable Pathway Signals (Custom)",
group=group_signals)

show_cisd = input.bool(true, "Enable Cisd Signals (Custom)",
group=group_signals)

show_hf_shapes= input.bool(false, "Show 5m Signals on Chart",
group=group_signals, tooltip="Plots the 5m timeframe signals visually on the
current timeframe chart.")
```

```
group_ema = "9 EMA Retrace"

show_ema = input.bool(true, "Enable 9 EMA Retrace Signals (Custom)",
group=group_ema)

emaFastLen = input.int(9, "Fast EMA", group=group_ema)

emaSlowLen = input.int(20, "Slow EMA", group=group_ema)

emaLineLength = input.int(25, "EMA Trade Line Length", minval = 1,
group=group_ema)

requireCross = input.bool(true, "Require EMA Cross Before Retrace",
group=group_ema)

filter_ema_hf = input.bool(true, "Filter EMA Signals by HTF Box (Custom)",
group=group_ema)
```

```
group_display = "Display Settings"

show_hf_box = input.bool(true, "Show 5m HTF Reading Box",
group=group_display)
```

```
show_recent_lines = input.bool(false, "Only Show Most Recent Trade Lines",  
group=group_display, tooltip="When enabled, previous entry/stop/target lines  
are removed when a new signal occurs.")
```

```
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)
```

```
// --- Mode Logic ---
```

```
is_1min_mode = trading_mode == "1 min 9 EMA with HTF Support"
```

```
is_5min_mode = trading_mode == "5 min Signals Only"
```

```
is_custom_mode = trading_mode == "Custom"
```

```
bool active_sweep = is_custom_mode ? show_sweep : is_5min_mode
```

```
bool active_pathway = is_custom_mode ? show_pathway : is_5min_mode
```

```

bool active_cisd    = is_custom_mode ? show_cisd : is_5min_mode

bool active_ema     = is_custom_mode ? show_ema : (is_1min_mode or
is_5min_mode)

bool active_ema_filter= is_custom_mode ? filter_ema_hrf : is_1min_mode


if only_ema

    active_sweep := false

    active_pathway := false

    active_cisd := false

    active_ema := true


if only_cisd

    active_sweep := false

    active_pathway := false

    active_ema := false

    active_cisd := true


// --- 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))

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

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

// Signal constraint

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
```

```
// --- CISD Logic ---
```

```
var float lastBearFVGTop = na
```

```
var float lastBullFVGBot = na
```

```
if high < low[2]
```

```
    lastBearFVGTop := low[2]
```

```
if low > high[2]
```

```
    lastBullFVGBot := high[2]
```

```
cisd_bull_cross = ta.crossover(close, lastBearFVGTop)
```

```
cisd_bear_cross = ta.crossunder(close, lastBullFVGBot)
```

```
var bool swept_key_h = false
```

```
var bool swept_key_l = false
```

```
// Check if any key level is swept
```

```
float[] all_h_levels = array.from(pdh, pwh, pmh, asia_h, lon_h, or_15_h,  
or_30_h, or_1h_h)
```

```
float swept_h_level = na
```

```
for lvl in all_h_levels
```

```
    if not na(lvl) and high >= lvl
```

```
        swept_key_h := true
```

```
        swept_h_level := lvl
```

```
float[] all_l_levels = array.from(pdl, pwl, pml, asia_l, lon_l, or_15_l, or_30_l,  
or_1h_l)
```

```
float swept_l_level = na
```

```
for lvl in all_l_levels
```

```
    if not na(lvl) and low <= lvl
```

```
        swept_key_l := true
```

```
        swept_l_level := lvl
```

```
bool closed_inside_h = not na(swept_h_level) and close < swept_h_level
```

```
bool closed_inside_l = not na(swept_l_level) and close > swept_l_level
```

```
bool bull_cisd_setup = swept_key_l and closed_inside_l and cisd_bull_cross  
and close > lastBearFVGTop
```

```
bool bear_cisd_setup = swept_key_h and closed_inside_h and  
cisd_bear_cross and close < lastBullFVGBot
```

```
if bull_cisd_setup
```

```
    swept_key_l := false
```

```
if bear_cisd_setup
```

```
swept_key_h := false
```

```
// --- HTF Signal Logic & Box (Calculated early for filtering) ---
```

```
[htf_ls, htf_ss, htf_lp, htf_sp, htf_bcisd, htf_bearcisd] =  
request.security(syminfo.tickerid, "5", [long_sweep_setup,  
short_sweep_setup, long_pathway_setup, short_pathway_setup,  
bull_cisd_setup, bear_cisd_setup])
```

```
var string last_htf_signal = "Awaiting 5m Signal"
```

```
var color last_htf_color = color.gray
```

```
var int last_htf_dir = 0 // 1 for bull, -1 for bear
```

```
if htf_bcisd
```

```
    last_htf_signal := "5m Bullish CISD"
```

```
    last_htf_color := color.green
```

```
    last_htf_dir := 1
```

```
else if htf_bearcisd
```

```
    last_htf_signal := "5m Bearish CISD"
```

```
    last_htf_color := color.red
```

```
    last_htf_dir := -1
```

```
else if htf_ls
```

```
    last_htf_signal := "5m Bullish Sweep"
```

```
    last_htf_color := color.green
```

```
    last_htf_dir := 1
```

```

else if htf_ss
    last_htf_signal := "5m Bearish Sweep"
    last_htf_color := color.red
    last_htf_dir := -1
else if htf_lp
    last_htf_signal := "5m Bullish Pathway"
    last_htf_color := color.green
    last_htf_dir := 1
else if htf_sp
    last_htf_signal := "5m Bearish Pathway"
    last_htf_color := color.red
    last_htf_dir := -1

var table htfTable = table.new(position.top_right, 1, 2,
    bgcolor=color.new(color.black, 70), border_color=color.gray, border_width=1)
if barstate.islast and show_htf_box
    table.cell(htfTable, 0, 0, "HTF Reading", text_color=color.white,
        text_size=size.small, bgcolor=color.new(color.blue, 70))
    table.cell(htfTable, 0, 1, last_htf_signal, text_color=last_htf_color,
        text_size=size.normal)

// --- Visualizing HTF Signals on Chart ---
plotshape(show_htf_shapes and htf_ls, "5m Long Sweep", shape.triangleup,
    location.belowbar, color.green, size=size.small, text="5m SW",
    textcolor=color.green)

```

```
plotshape(show_htf_shapes and htf_ss, "5m Short Sweep",  
shape.triangledown, location.abovebar, color.red, size=size.small, text="5m  
SW", textcolor=color.red)
```

```
plotshape(show_htf_shapes and htf_lp, "5m Long Pathway", shape.arrowup,  
location.belowbar, color.lime, size=size.small, text="5m PW",  
textcolor=color.lime)
```

```
plotshape(show_htf_shapes and htf_sp, "5m Short Pathway",  
shape.arrowdown, location.abovebar, color.maroon, size=size.small,  
text="5m PW", textcolor=color.maroon)
```

```
plotshape(show_htf_shapes and htf_bcisd, title="5m Bullish CISD",  
text="CISD", style=shape.labelup, location=location.belowbar,  
color=#089981, textcolor=color.white, size=size.small)
```

```
plotshape(show_htf_shapes and htf_bearcisd, title="5m Bearish CISD",  
text="CISD", style=shape.labeldown, location=location.abovebar,  
color=#f23645, textcolor=color.white, size=size.small)
```

```
// --- 9 EMA Retrace Logic ---
```

```
emaFast = ta.ema(close, emaFastLen)
```

```
emaSlow = ta.ema(close, emaSlowLen)
```

```
var int longState = 0
```

```
var int shortState = 0
```

```

if ta.crossover(emaFast, emaSlow)
    longState := 1
    shortState := 0
else if ta.crossunder(emaFast, emaSlow)
    shortState := 1
    longState := 0
else if not requireCross
    if emaFast > emaSlow and longState == 0
        longState := 1
    if emaFast < emaSlow and shortState == 0
        shortState := 1

if longState == 1 and low <= emaFast
    longState := 2
if shortState == 1 and high >= emaFast
    shortState := 2

ema_longSignal = longState == 2 and close > emaFast and active_ema
ema_shortSignal = shortState == 2 and close < emaFast and active_ema

// Apply HTF Box Filter if enabled
if active_ema_filter
    if last_htf_dir != 1

```

```

    ema_longSignal := false
    if last_htf_dir != -1
        ema_shortSignal := false

    if ema_longSignal
        longState := 0
    if ema_shortSignal
        shortState := 0

    plot(active_ema ? emaFast : na, "Fast EMA", color = color.blue)
    plot(active_ema ? emaSlow : na, "Slow EMA", color = color.red)

    // --- Execution & Visualizing Entries & Lines ---
    bool final_long_sweep  = long_sweep_setup and active_sweep
    bool final_short_sweep = short_sweep_setup and active_sweep

    bool final_long_pathway = long_pathway_setup and active_pathway
    bool final_short_pathway = short_pathway_setup and active_pathway

    bool final_bull_cisd  = bull_cisd_setup and active_cisd
    bool final_bear_cisd  = bear_cisd_setup and active_cisd

```

```
bool any_long_signal = final_long_sweep or final_long_pathway or  
final_bull_cisd
```

```
bool any_short_signal = final_short_sweep or final_short_pathway or  
final_bear_cisd
```

```
plotshape(final_long_sweep, "Long Sweep Entry", shape.triangleup,  
location.belowbar, color.green, size=size.small)
```

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

```
plotshape(final_long_pathway, "Long Pathway Entry", shape.arrowup,  
location.belowbar, color.lime, size=size.small)
```

```
plotshape(final_short_pathway, "Short Pathway Entry", shape.arrowdown,  
location.abovebar, color.maroon, size=size.small)
```

```
plotshape(final_bull_cisd, title="Bullish Cisd", text="Cisd",  
style=shape.labelup, location=location.belowbar, color=#089981,  
textcolor=color.white, size=size.small)
```

```
plotshape(final_bear_cisd, title="Bearish Cisd", text="Cisd",  
style=shape.labeldown, location=location.abovebar, color=#f23645,  
textcolor=color.white, size=size.small)
```

```
plotshape(ema_longSignal, "Long EMA Entry", shape.triangleup,  
location.belowbar, color.new(color.green, 0), size = size.small)
```

```
plotshape(ema_shortSignal, "Short EMA Entry", shape.triangledown,  
location.abovebar, color.new(color.red, 0), size = size.small)
```

```
// State variables to hold recent lines
```

```
var line last_ep_line = na
```

```
var line last_sl_line = na
```

```
var line last_tp_line = na
```

```
draw_lines(float ep, float sl, float tp, int length, line in_ep, line in_sl, line in_tp)
```

```
=>
```

```
  if show_recent_lines
```

```
    line.delete(in_ep)
```

```
    line.delete(in_sl)
```

```
    line.delete(in_tp)
```

```
    out_ep = line.new(bar_index, ep, bar_index + length, ep, color = color.gray,  
style = line.style_dotted, width = 2)
```

```
    out_sl = line.new(bar_index, sl, bar_index + length, sl, color = color.red, style  
= line.style_dotted, width = 2)
```

```
    out_tp = line.new(bar_index, tp, bar_index + length, tp, color = color.green,  
style = line.style_dotted, width = 2)
```

```
    [out_ep, out_sl, out_tp]
```

```
if final_bull_cisd
```

```
  float ep = close
```

```
  float sl = low
```

```
  float risk = ep - sl
```

```
float tp = ep + (risk * rrRatio)

if risk > 0

    [new_ep, new_sl, new_tp] = draw_lines(ep, sl, tp, 10, last_ep_line,
last_sl_line, last_tp_line)

    last_ep_line := new_ep

    last_sl_line := new_sl

    last_tp_line := new_tp
```

```
if final_bear_cisd

    float ep = close

    float sl = high

    float risk = sl - ep

    float tp = ep - (risk * rrRatio)

    if risk > 0

        [new_ep, new_sl, new_tp] = draw_lines(ep, sl, tp, 10, last_ep_line,
last_sl_line, last_tp_line)

        last_ep_line := new_ep

        last_sl_line := new_sl

        last_tp_line := new_tp
```

```
if ema_longSignal

    float ep = close

    float sl = emaSlow

    float risk = ep - sl
```

```

float tp = ep + (risk * rrRatio)

if risk > 0

    [new_ep, new_sl, new_tp] = draw_lines(ep, sl, tp, emaLineLength,
last_ep_line, last_sl_line, last_tp_line)

    last_ep_line := new_ep

    last_sl_line := new_sl

    last_tp_line := new_tp


if ema_shortSignal

    float ep = close

    float sl = emaSlow

    float risk = sl - ep

    float tp = ep - (risk * rrRatio)

    if risk > 0

        [new_ep, new_sl, new_tp] = draw_lines(ep, sl, tp, emaLineLength,
last_ep_line, last_sl_line, last_tp_line)

        last_ep_line := new_ep

        last_sl_line := new_sl

        last_tp_line := new_tp


// Move AOI labels to the right

string score_l_str = str.toString(score_l)

string score_h_str = str.toString(score_h)

```

```
if final_long_sweep and score_l > 0 and show_aoi_score
```

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

```
if final_short_sweep and score_h > 0 and show_aoi_score
```

```
    label.new(bar_index + 1, high, "AOI Score: " + score_h_str,  
    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  $\leq$  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 ---
```

```
alertcondition(any_long_signal or any_short_signal or ema_longSignal or  
ema_shortSignal, title="MFB - Coherence", message="MFB Coherence signal  
triggered on {{ticker}}")
```

```
bool ema_htf_agreement = (ema_longSignal and last_htf_dir == 1) or  
(ema_shortSignal and last_htf_dir == -1)
```

```
alertcondition(ema_htf_agreement, title="MFB - 9 EMA & HTF Agreement",  
message="EMA Retrace & HTF in agreement")
```