Hospital Scraper Pattern Reference Guide

Complete Edition - All 9 Patterns

Pattern Selection Decision Tree

Are name and title in same element?

├─ YES → Use "combined_h2" (Pattern 2)

└─ NO → Continue...

Is it a table structure?

├─ YES →

│ ├─ Simple columns? → Use "table_rows" (Pattern 3)

│ └─ Nested elements in cells? → Use "custom_table_nested" (Pattern 8)

└─ NO → Continue...

Are they in list items (ul/ol)?

├─ YES → Use "list_items" (Pattern 6)

└─ NO → Continue...

Do elements have specific CSS classes?

├─ YES → Use "div_classes" (Pattern 5)

└─ NO → Continue...

Is it a gallery/card layout?

├─ YES → Use "boardcard_gallery" (Pattern 7)

└─ NO → Continue...

Are they sequential P elements with spacers?

├─ YES → Use "sequential_p_with_spacers" (Pattern 9)

└─ NO → Continue...

Is it specifically H2→P structure?

├─ YES → Use "h2_name_p_title" (Pattern 4)

└─ NO → Use "h2_name_h3_title" (Pattern 1) with custom elements

---

Pro Tips

1. **Most flexible patterns:** Pattern 1 (h2_name_h3_title) and Pattern 2 (combined_h2) - can adapt to many HTML structures by customizing element types
2. **When in doubt:** Start with Pattern 1 and customize the element types to match what you see in the HTML
3. **Complex structures:** Use Pattern 8 (custom_table_nested) with CSS selectors for maximum control
4. **Missing people:** Works with ALL patterns - just add missing_people section to YAML
5. **Test before committing:** Always use helper$test_hospital_config() or quick_test(FAC) before adding to main YAML
6. **Use the helper tools:**
    - helper$analyze_hospital_structure(FAC, name, url) - Inspects HTML and suggests patterns
    - helper$show_pattern_guide() - Shows pattern identification guide
    - quick_test(FAC) - Quick test of single hospital from YAML
7. **Debugging workflow:**
    - Inspect the page HTML manually (Right-click → Inspect)
    - Use helper$analyze_hospital_structure() to get suggestions
    - Test with quick_test(FAC)

- If it fails, adjust the pattern or add missing_people
- Re-test until successful

Pattern 1: h2_name_h3_title (Sequential Elements)

Description: Names and titles are in separate, sequential HTML elements. Name element is followed by title element.

YAML Structure:

pattern: "h2_name_h3_title"

html_structure:

name_element: "h2" # ← CUSTOMIZABLE

title_element: "h3" # ← CUSTOMIZABLE

Customizable Components:

- **name_element:** Any heading tag (h1, h2, h3, h4, h5, h6), p, span, div, strong
- **title_element:** Any heading tag (h1, h2, h3, h4, h5, h6), p, span, div, strong

Examples:

H3 names, H4 titles

name_element: "h3"

title_element: "h4"

H2 names, P titles

name_element: "h2"

title_element: "p"

Strong names, span titles

name_element: "strong"

title_element: "span"

When to Use:

Names and titles are in different element types, appearing sequentially on the page.

Real Examples:

- FAC-707: Ross Memorial Hospital
- FAC-624: Campbellford Memorial Hospital
- FAC-596: Alliston Stevenson Memorial Hospital

Pattern 2: combined_h2 (Combined Name+Title)

Description: Name and title are in the SAME element, separated by a character/string.

YAML Structure:

```
pattern: "combined_h2"

html_structure:

 combined_element: "h2"  # ← CUSTOMIZABLE

 separator: " - "       # ← CUSTOMIZABLE
```

Customizable Components:

- **combined_element:** Any HTML element (h1, h2, h3, h4, p, div, span, li, td)
- **separator:** Any string that separates name from title

Examples:

```
# H3 with dash separator

combined_element: "h3"

separator: " - "


# P with comma separator

combined_element: "p"

separator: ", "
```

List items with pipe separator

combined_element: "li"

separator: " | "

Div with colon

combined_element: "div"

separator: ": "

When to Use:

Name and title appear together in one element like "John Smith - CEO" or "Jane Doe, President"

Real Examples:

- FAC-941: Humber River Hospital (h3 with " - " separator)
- FAC-952: Lakeridge Health (h3 with ", " separator)

Pattern 3: table_rows (Table Structure)

Description: Names and titles are in table cells, typically in different columns.

YAML Structure:

pattern: "table_rows"

html_structure:

structure_type: "table"

name_location: "td_column_1" # ← CUSTOMIZABLE (column number)

title_location: "td_column_2" # ← CUSTOMIZABLE (column number)

Customizable Components:

- **name_location:** "td_column_X" where X is the column number (1, 2, 3, etc.)
- **title_location:** "td_column_X" where X is the column number (1, 2, 3, etc.)

Examples:

Name in column 1, title in column 2

name_location: "td_column_1"

title_location: "td_column_2"

Name in column 2, title in column 3

name_location: "td_column_2"

title_location: "td_column_3"

Reversed order

name_location: "td_column_2"

title_location: "td_column_1"

When to Use:

Executives are listed in an HTML table with clear column structure.

Real Examples:

- FAC-661: Cambridge Memorial Hospital

Pattern 4: h2_name_p_title (Specific H2→P Pattern)

Description: Name in H2 element, title in the immediately following P element. More strict than Pattern 1.

YAML Structure:

pattern: "h2_name_p_title"

html_structure:

name_element: "h2" # Fixed as h2

title_element: "p" # Fixed as p

Customizable Components:

- **Not customizable** - this is a specific pattern for H2→P structure

- If you need different elements, use Pattern 1 instead

When to Use:

Specifically when you have <h2>Name</h2> followed by Title structure.

Real Examples:

- FAC-953: Sunnybrook Health Sciences Centre

Pattern 5: div_classes (CSS Class-Based)

Description: Names and titles are in elements with specific CSS classes.

YAML Structure:

pattern: "div_classes"

html_structure:

name_class: "staff-name" # ← CUSTOMIZABLE

title_class: "staff-title" # ← CUSTOMIZABLE

container_class: "staff-member" # ← OPTIONAL

Customizable Components:

- **name_class:** CSS class name for name elements (without the dot)
- **title_class:** CSS class name for title elements (without the dot)
- **container_class:** (Optional) Parent container class

Examples:

Standard div classes

name_class: "executive-name"

title_class: "executive-title"

Span classes

name_class: "bio-name"

title_class: "bio-position"

Card-based layout

name_class: "card-title"

title_class: "card-subtitle"

container_class: "team-card"

When to Use:

HTML uses semantic CSS classes like class="name" and class="title".

Real Examples:

- FAC-905: Oak Valley Health
- FAC-606: Barrie Royal Victoria Regional HC
- FAC-979: Toronto Scarborough Health Network

Pattern 6: list_items (List-Based)

Description: Names and titles are in list items (ul/ol), either combined or sequential.

YAML Structure:

pattern: "list_items"

html_structure:

list_type: "ul" # ← CUSTOMIZABLE (ul or ol)

item_element: "li" # Fixed as li

format: "combined" # ← CUSTOMIZABLE (combined or sequential)

separator: " - " # ← CUSTOMIZABLE (if combined)

Customizable Components:

- **list_type:** "ul" (unordered) or "ol" (ordered)
- **format:** "combined" (name-title in same li) or "sequential" (separate li elements)
- **separator:** If combined format, the separator string

Examples:

Combined list items with dash

list_type: "ul"

format: "combined"

separator: " - "

Combined list items with comma

list_type: "ul"

format: "combined"

separator: ", "

Sequential list items (name li, then title li)

list_type: "ul"

format: "sequential"

When to Use:

Executives are in <ul> or <ol> lists.

Real Examples:

- FAC-957: Belleville Quinte Health Care (comma separator)

Pattern 7: boardcard_gallery (Special Gallery Pattern)

Description: Executives in card/gallery layout with specific div class, name and title separated by comma.

YAML Structure:

pattern: "boardcard_gallery"

html_structure:

container_class: "boardcard" # ← CUSTOMIZABLE

text_format: "name_comma_title"

separator: "," # ← CUSTOMIZABLE

Customizable Components:

- **container_class:** CSS class of the card/gallery container
- **separator:** Character separating name from title (usually comma)

Examples:

Standard boardcard

container_class: "boardcard"

separator: ","

Different gallery class

container_class: "executive-card"

separator: " | "

Team member cards

container_class: "team-member"

separator: " - "

When to Use:

Gallery or card-based layouts with combined name-title text.

Real Examples:

- FAC-935: Thunder Bay Regional Health Sciences Centre

Pattern 8: custom_table_nested (Complex Nested Tables)

Description: Table structure with nested elements inside cells (like p inside td, div inside td).

YAML Structure:

pattern: "custom_table_nested"

```
html_structure:
  structure_type: "table_with_nested_elements"
  name_selector: "td p[style*='text-align: left']"    # ← CUSTOMIZABLE
  title_selector: "td div[style*='text-align: left']" # ← CUSTOMIZABLE
  container: "td"                                     # ← CUSTOMIZABLE
```

Customizable Components:

- **name_selector:** Full CSS selector for name element (can include attributes, styles)
- **title_selector:** Full CSS selector for title element
- **container:** Parent element containing both name and title

Examples:

```
# Names in p, titles in div with style attribute
name_selector: "td p[style*='text-align: left']"
title_selector: "td div[style*='text-align: left']"

# Names in span with class, titles in div
name_selector: "td span.executive-name"
title_selector: "td div.executive-title"

# Different container
name_selector: "div.profile p.name"
title_selector: "div.profile span.title"
container: "div.profile"
```

When to Use:

Complex table structures with nested HTML elements and specific styling.

Real Examples:

- FAC-777: Queensway Carleton Hospital

Pattern 9: sequential_p_with_spacers (Sequential P Elements)

Description: Names and titles are in sequential elements with empty spacers between each executive. Pattern skips empty elements and pairs name P with next title P.

YAML Structure:

pattern: "sequential_p_with_spacers"

html_structure:

name_element: "p"

title_element: "p"

notes: "Sequential P elements: name P, title P, empty P spacer, repeat"

Customizable Components:

- **Not customizable** - specifically designed for P element sequences
- Automatically handles empty P spacers

How it Works:

1. Reads all elements in order
2. Identifies names using pattern matching
3. Looks ahead for next non-empty P (title)
4. Validates title matches executive patterns
5. Pairs name with title
6. Skips empty P elements automatically

When to Use:

When executives are in sequential tags with this pattern:

John Smith

Chief Executive Officer

Jane Doe

Chief Operating Officer

Real Examples:

- FAC-932: Elizabeth Bruyere Hospital

Missing People Feature

All patterns support manually adding executives who don't appear in the main HTML structure.

YAML Structure:

```
html_structure:
  [pattern-specific settings]
  missing_people:
    - name: "Dr. John Smith"
      title: "President and CEO"
    - name: "Jane Doe"
      title: "Chief of Staff"
```

When to Use:

- CEO or senior leaders listed separately on the page
- Executives whose HTML structure differs from the rest
- People mentioned but not in the main scraping structure

Works With:

ALL 9 patterns - just add the missing_people section to any hospital configuration

---

Inspection Checklist

When analyzing a new hospital website:

1. **Right-click → Inspect Element** on executive names

2. **Note the HTML tag** (h1, h2, h3, p, div, span, td, li)
3. **Check for CSS classes or IDs**
4. **See if names and titles are in same or different elements**
5. **If separate, what's the relationship?** (next sibling, same parent, table row)
6. **Look for consistent patterns across all executives**
7. **Note any missing people not in the main structure**
8. **Use helper tools:**
9. helper$analyze_hospital_structure(FAC, "Hospital Name", "URL")

Testing Workflow

1. **Add hospital to YAML** with best-guess pattern
2. **Test configuration:**
3. quick_test(FAC_NUMBER)
4. **Review results** - did it find the expected number?
5. **If failed:**
    - Use helper$analyze_hospital_structure() for more details
    - Adjust pattern or element selectors
    - Add missing_people if needed
    - Test again
6. **Once working, update status to "configured"**

Common Patterns Summary

Pattern	Best For	Flexibility
1. h2_name_h3_title	Sequential different elements	High - customizable elements

Pattern	Best For	Flexibility
2. combined_h2	Name+title in one element	High - customizable separator
3. table_rows	Simple table columns	Medium - column numbers
4. h2_name_p_title	Specific H2→P only	Low - fixed structure
5. div_classes	CSS class-based	High - any class names
6. list_items	ul/ol lists	Medium - combined or sequential
7. boardcard_gallery	Gallery/card layouts	Medium - specific to cards
8. custom_table_nested	Complex nested tables	Very High - full CSS selectors
9. sequential_p_with_spacers	Sequential P with spacers	Low - specific to P elements

---

Quick Reference Commands

```
# Load everything
setwd("E:/ExecutiveSearchYaml/code/")
source("pattern_based_scraper.R")
source("quick_test_single.R")
source("hospital_configuration_helper.R")

# Test single hospital
quick_test(FAC_NUMBER)

# Test multiple hospitals
quick_test_batch(c(707, 935, 941, 952))
```

```
# Analyze HTML structure
helper$analyze_hospital_structure(FAC, "Hospital Name", "URL")

# Show all available hospitals
show_available_facs()

# Test all configured hospitals
results <- test_all_hospitals_from_yaml()
```

Troubleshooting Guide

No results found?

1. Check if URL is accessible
2. Verify pattern matches HTML structure
3. Use helper$analyze_hospital_structure() for detailed inspection
4. Try different pattern or customize element selectors

Wrong names/titles extracted?

1. Check name/title patterns in YAML config
2. Verify separator (for combined patterns)
3. Look for phone extensions or extra text to clean
4. Add custom cleaning rules if needed

Missing some executives?

1. Check expected_executives count
2. Look for executives in different HTML structure
3. Add them to missing_people section
4. Verify all executives on page are actually captured

Pattern not recognized?

1. Verify pattern name matches exactly (case-sensitive)
2. Check pattern is in switch statement in scraper
3. Ensure pattern function is defined
4. Test with quick_test(FAC) for error messages

Last Updated: October 2, 2025 Version: 2.0 - Complete 9-Pattern Edition