OTTL Quick Reference

A compact cheatsheet for day-to-day use.

Core Syntax

Statements are composed of a function and an optional condition.

```
function(arguments) where condition
```

Data Types

- **String:** `"hello"`
- **Integer:** `42`
- **Float:** `3.14`
- **Boolean:** `true` / `false`
- **Nil:** `nil`
- **Byte Slice:** `0x48656c6c6f`
- **List:** `["a", "b"]`
- **Map:** `{"key": "value"}`

Operators

Type	Operators
Comparison	`==` `!=` `>` `>=` `<` `<=`
Logical	`and` `or` `not`
Math	`+` `-` `*` `/` `()`

Precedence: 1. `()`, 2. `not`, 3. `* /`, 4. `+ -`, 5. `and`, 6. `or`.

Contexts & Path Shortcuts

Resource Context

```
resource.attributes["key"]
resource.cache["key"]
```

Span Context

```
span.name
span.status.code
span.attributes["key"]
span.start_time / span.end_time
```

Log Context

```
log.body / log.body.string
log.severity_text
log.attributes["key"]
log.trace_id.string
```

Metric/DataPoint Context

```
metric.name / metric.unit
datapoint.value_double / .value_int
datapoint.attributes["key"]
```

OTTL Quick Reference

Essential Functions & Common Patterns

Essential Functions

Editors (lowercase)

`set(target, value)`

`delete_key(target, "key")`

`delete_matching_keys(target, ".*secret.*")`

`keep_keys(target, ["key1", "key2"])`

`replace_match(target, "pattern", "replacement")`

`replace_all_matches(target, "pattern", "replacement")`

`limit(target, max_elements)`

`flatten(target)`

Converters (Uppercase)

`IsMatch(target, "pattern")`

`ParseJSON(target)`

`ParseKeyValue(target, "=", "&")`

`Concat(["val1", "val2"], " ")`

`Split(target, ",")`

`Substring(target, start, length)`

`ToUpperCase(target)` / `ToLowerCase(target)`

`String(target)` / `Int(target)`

Common Patterns & Snippets

PII Redaction: Replace credit card numbers in the log body.

```
replace_all_matches(log.body, "\\b(?:\\d{4}[\\s-]?){3}\\d{4}\\b", "****-****-****-****")
```

HTTP Normalization: Set span status to error for 4xx/5xx codes.

```
set(span.status.code, STATUS_CODE_ERROR) where Int(span.attributes["http.status_code"]) >= 400
```

Metric Unit Conversion: Convert milliseconds to seconds.

```
set(datapoint.value_double, datapoint.value_double / 1000) where metric.unit == "ms"
```

OTTL Quick Reference

Regex, Troubleshooting & Best Practices

Common Regex Patterns

Remember to use double backslashes (`\\d`) in OTTL strings.

Email	`"[^@]+@[^@]+\\.[^@]+"`
IP Address	`"\\b(\\d{1,3}\\.){3}\\d{1,3}\\b"`
UUID	`"[0-9a-f]{8}-([0-9a-f]{4}-){3}[0-9a-f]{12}"`
URL	`"https?://[^\\s]+"`

Troubleshooting Checklist

- **Syntax:** Check for balanced parentheses, quotes, and valid operator spelling.
- **Type Errors:** Validate data types with `IsString` , `IsInt` etc. before conversion.
- **Nil Values:** Always check for nil before accessing fields (e.g., `where field ≠ nil`).
- **Logic Errors:** Add temporary debug attributes with `set(span.attributes["debug"], ...)` to trace values.

Safety & Performance Best Practices

DOs

- Order conditions from cheapest to most expensive.
- Use early filtering to reduce processing.
- Cache expensive results: `set(resource.cache["key"], ...)` .
- Validate inputs before complex processing.

DON'Ts

- Don't repeat expensive function calls.
- Avoid complex regex on very large strings.
- Don't process unnecessary data; filter it out first.
- Don't forget to handle potential nil values.