```
# MZ2POL-05: Segments Implementation

> **Series:** MZ2POL | **Notebook:** 6 of 8 | **Created:** December 2025

## Overview

This notebook provides a comprehensive guide to **Segments** - the DQL-
powered data filtering mechanism that replaces Management Zone filtering.
Segments allow you to create reusable, dynamic filter conditions for cross-
app data segmentation.

## Prerequisites

- Completed MZ2POL-01 through MZ2POL-04
- Understanding of DQL basics
- Knowledge of your MZ filtering patterns

## Learning Objectives

By the end of this notebook, you will:
1. Understand how Segments work with Grail
2. Be able to create Segments that replace MZ filtering
3. Know how to use variables for dynamic filtering
4. Understand Segment application across apps

---

## 1. What Are Segments?

### Definition

**Segments** are reusable, pre-defined filter conditions powered by DQL. They
provide query-time filtering to control what data users see.

### Key Characteristics

| Feature | Description |
|---------|-------------|
| **DQL-powered** | Full query language flexibility |
| **Query-time** | Evaluated at query execution, not precalculated |
| **Multi-dimensional** | Can be layered for precise filtering |
| **Dynamic** | Support variables for runtime flexibility |
| **Cross-app** | Work across all Grail-based apps |

### Segments vs. Management Zones

| Aspect | Management Zone | Segment |
```

```
|--------|----------------|----------|
| Evaluation | Precalculated | Query-time |
| Performance | Bottleneck at scale | Highly scalable |
| Flexibility | Entity rules only | Any DQL filter |
| Variables | Not supported | Supported |
| Multi-dimensional | No | Yes |
| Grail support | No | Yes |

---

## 2. How Segments Work

### Query-Time Injection

![Segment Injection]
(
ciIHZpZXdCb3g9IjAgMCA4MDAgMzAwIj4KICA8ZGVmcz4KICAgIDxsaW5lYXJHcmFkaWVudCBpZD0
icXVlcnlHcmFkIiB4MT0iMCUiIHkxPSIwJSIgeDI9IjEwMCUiIHkyPSIxMDAlIj4KICAgICAgPHN0
b3Agb2Zmc2V0PSIwJSIgc3R5bGU9InN0b3AtY29sb3I6IzNiODJmNjtzdG9wLW9wYWNpdHk6MSIgL
z4KICAgICAgPHN0b3Agb2Zmc2V0PSIxMDAlIiBzdHlsZT0ic3RvcC1jb2xvcjojMjU2M2ViO3N0b3
Atb3BhY2l0eToxIiAvPgogICAgPC9saW5lYXJHcmFkaWVudD4KICAgIDxsaW5lYXJHcmFkaWVudCB
pZD0ic2VnR3JhZCIgeDE9IjAlIiB5MT0iMCUiIHgyPSIxMDAlIiB5Mj0iMTAwJSI+CiAgICAgIDxz
dG9wIG9mZnNldD0iMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiMxMGI5ODE7c3RvcC1vcGFjaXR5OjEiI
C8+CiAgICAgIDxzdG9wIG9mZnNldD0iMTAwJSIgc3R5bGU9InN0b3AtY29sb3I6IzA1OTY2OTtzdG
9wLW9wYWNpdHk6MSIgLz4KICAgIDwvbGluZWFyR3JhZGllbnQ+CiAgICA8bGluZWFyR3JhZGllbnQ
gaWQ9ImVmZmVjdGl2ZUdyYWQiIHgxPSIwJSIgeTE9IjAlIiB4Mj0iMTAwJSIgeTI9IjEwMCUiPgog
ICAgICA8c3RvcCBvZmZzZXQ9IjAlIiBzdHlsZT0ic3RvcC1jb2xvcjojOGI1Y2Y2O3N0b3Atb3BhY
2l0eToxIiAvPgogICAgICA8c3RvcCBvZmZzZXQ9IjEwMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiM3Yz
NhZWQ7c3RvcC1vcGFjaXR5OjEiIC8+CiAgICA8L2xpbmVhckdyYWRpZW50PgogICAgPGxpbmVhckdy
YWRpZW50IGlkPSJyZXN1bHRHcmFkIiB4MT0iMCUiIHkxPSIwJSIgeDI9IjEwMCUiIHkyPSIxMDAl
Ij4KICAgICAgPHN0b3Agb2Zmc2V0PSIwJSIgc3R5bGU9InN0b3AtY29sb3I6I2Y1OWUwYjtzdG9wL
W9wYWNpdHk6MSIgLz4KICAgICAgPHN0b3Agb2Zmc2V0PSIxMDAlIiBzdHlsZT0ic3RvcC1jb2xvcj
ojZDk3NzA2O3N0b3Atb3BhY2l0eToxIiAvPgogICAgPC9saW5lYXJHcmFkaWVudD4KICAgIDxmaWx
0ZXIgaWQ9InNlZ1NoYWRvdyI+CiAgICAgIDxmZURyb3BTaGFkb3cgZHg9IjEiGR5PSIxIiBzdGRE
ZXZpYXRpb249IjIiIGZsb29kLW9wYWNpdHk9IjAuMTUiLz4KICAgIDwvZmlsdGVyPgogICAgPG1hc
mtlciBpZD0ic2VnQXJyb3ciIG1hcmtlcldpZHRoPSIxMCIgbWFya2VySGVpZ2h0PSI3IiByZWZYPS
I5IiByZWZZPSIzLjUiIG9yaWVudD0iYXV0byI+CiAgICAgIDxwb2x5Z29uIHBvaW50cz0iMCAwLCA
xMCAzLjUsIDAgNyIgZmlsbD0iIzY0NzQ4YiIvPgogICAgPC9tYXJrZXI+CiAgPC9kZWZzPgoKICA8
IS0tIEJhY2tncm91bmQgLS0+CiAgPHJlY3Qgd2lkdGg9IjgwMCIgaGVpZ2h0PSIzMDAiIGZpbGw9I
iNmOGY5ZmEiIHJ4PSIxMCIvPgoKICA8IS0tIFRpdGxlIC0tPgogIDx0ZXh0IHg9IjQwMCIgeT0iMj
giIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxOCIgZm9udC13ZWl
naHQ9ImJvbGQiIGZpbGw9IiMzMzMiIHRleHQtYW5jaG9yPSJtaWRkbGUiPlNlZ21lbnQgUXVlcnkt
VGltZSBJbmplY3Rpb248L3RleHQ+CiAgPHRleHQgeD0iNDAwIiB5PSI0OCIgZm9udC1mYW1pbHk9I
kFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEyIiBmaWxsPSIjNjY2IiB0ZXh0LWFuY2hvcj
0ibWlkZGxlIj5TZWdtZW50cyBpbmplY3QgRFFMIGZpbHRlcnMgYXQgcXVlcnkgZXhlY3V0aW9uICh
ub3QgcHJlY2FsY3VsYXRlZCBsaWtlIE1acyk8L3RleHQ+CgogIDwhLS0gQ29uZmlndXJhdGlvbiBi
LS0+CiAgPHJlY3QgeD0iNDAiIHk9IjcwIiB3aWR0aD0iMTcwIiBoZWlnaHQ9IjgwIiBmaWxsPSIjO
mlsbD0idXJsKCNxdWVyeUdyYWQpIiBmaWx0ZXI9InVybCgjc2VnU2hhZG93KSIvPgogIDx0ZXh0IH
```

```
g9IjExNSIgeT0iOTUiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSI
xMSIgZm9udC13ZWlnaHQ9ImJvbGQiIGZpbGw9IndoaXRlIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5V
c2VyIFF1ZXJ5PC90ZXh0PgogIDx0ZXh0IHg9IjExNSIgeT0iMTE4IiBmb250LWZhbWlseT0ibW9ub
3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTUsMjU1LDAuOSkiIHRleHQtYW
5jaG9yPSJtaWRkbGUiPmZldGNoIGxvZ3M8L3RleHQ+CiAgPHRleHQgeD0iMTE1IiB5PSIxMzMiIGZ
vbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9InJnYmEoMjU1LDI1NSwy
NTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+fCBsaW1pdCAxMDA8L3RleHQ+CgogIDwhLS0gU
Gx1cyBzaWduIC0tPgogIDx0ZXh0IHg9IjExNSIgeT0iMTE1IiBmb250LWZhbWlseT0iQXJpYWwsIH
NhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMjQiIGZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSIjNjQ3NDh
iIj4rPC90ZXh0PgoKICA8IS0tIEFjdGl2ZSBTZWdtZW50IEJveCAtLT4KICA8cmVjdCB4PSIyNDAi
IHk9IjcwIiB3aWR0aD0iMTcwIiBoZWlnaHQ9IjgwIiByeD0iOCIgZmlsbD0idXJsKCNzZWdHcmFkK
SIgZmlsdGVyPSJ1cmwoI3NlZ1NoYWRvdykiLz4KICA8dGV4dCB4PSIzMjUiIHk9IjEwNiIgZm9udC
1zaXplPSIxMCIgZm9udC13ZWlnaHQ9IjYwMCIgZmlsbD0icmdiYSgyNTUsMjU1LDI1NSwwLjkpIi
ZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSJyZ2JhKDI
1NSwyNTUsMjU1LDAuOSkiIHRleHQtYW5jaG9yPSJtaWRkbGUiPnwgIDx0ZXh0IHg9IjI3NSIgeT0i
MTEiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9InJnYmEoMjU1
LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+dGFnczogIDx0ZXh0IHg9IjI3NSIg
eT0iMjciIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9InJnYmEo
MjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+dGFnczogIDx0ZXh0IHg9IjI3
NSIgeT0iNDMiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9InJn
YmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+dGFnczogIDx0ZXh0IHg9
IjI3NSIgeT0iNTkiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9
InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+dGFnczogIDx0ZXh0
IHg9IjI3NSIgeT0iNzUiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZp
bGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+dGFnczogIDx0
ZXh0IHg9IjI3NSIgeT0iOTEiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAi
IGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+dGFnczog
IDx0ZXh0IHg9IjI3NSIgeT0iMTA3IiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9
IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTUsMjU1LDAuOSkiIHRleHQtYW5jaG9yPSJtaWRkbGUiPmZl
dGNoIGxvZ3M8L3RleHQ+CiAgPHRleHQgeD0iMTE1IiB5PSIxMjMiIGZvbnQtZmFtaWx5PSJtb25v
c3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1h
bmNob3I9Im1pZGRsZSI+dGFnczogIDx0ZXh0IHg9IjExNSIgeT0iMTM5IiBmb250LWZhbWlseT0i
bW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTUsMjU1LDAuOSkiIHRl
eHQtYW5jaG9yPSJtaWRkbGUiPmZldGNoIGxvZ3M8L3RleHQ+CiAgPHRleHQgeD0iMTE1IiB5PSIx
NTUiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9InJnYmEoMjU1
LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+dGFnczogIDx0ZXh0IHg9IjExNSIg
eT0iMTcxIiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSJyZ2Jh
KDI1NSwyNTUsMjU1LDAuOSkiIHRleHQtYW5jaG9yPSJtaWRkbGUiPmZldGNoIGxvZ3M8L3RleHQ+
CiAgPHRleHQgeD0iMTE1IiB5PSIxODciIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6
ZT0iMTAiIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+
dGFnczogIDx0ZXh0IHg9IjExNSIgeT0iMjAzIiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250
LXNpemU9IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTUsMjU1LDAuOSkiIHRleHQtYW5jaG9yPSJtaWRk
bGUiPmZldGNoIGxvZ3M8L3RleHQ+CiAgPHRleHQgeD0iMTE1IiB5PSIyMTkiIGZvbnQtZmFtaWx5
PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIg
dGV4dC1hbmNob3I9Im1pZGRsZSI+dGFnczogIDx0ZXh0IHg9IjExNSIgeT0iMjM1IiBmb250LWZh
bWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTUsMjU1LDAu
OSkiIHRleHQtYW5jaG9yPSJtaWRkbGUiPmZldGNoIGxvZ3M8L3RleHQ+CiAgPHRleHQgeD0iMTE1
IiB5PSIyNTEiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9InJn
YmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+dGFnczogIDx0ZXh0IHg9
IjExNSIgeT0iMjY3IiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxs
PSJyZ2JhKDI1NSwyNTUsMjU1LDAuOSkiIHRleHQtYW5jaG9yPSJtaWRkbGUiPmZldGNoIGxvZ3M8
L3RleHQ+CiAgPHRleHQgeD0iMTE1IiB5PSIyODMiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZv
bnQtc2l6ZT0iMTAiIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1p
ZGRsZSI+dGFnczogIDx0ZXh0IHg9IjExNSIgeT0iMjk5IiBmb250LWZhbWlseT0ibW9ub3NwYWNl
IiBmb250LXNpemU9IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTUsMjU1LDAuOSkiIHRleHQtYW5jaG9y
PSJtaWRkbGUiPmZldGNoIGxvZ3M8L3RleHQ+CiAgPHRleHQgeD0iMTE1IiB5PSIzMTUiIGZvbnQt
ZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUs
MC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+dGFnczogIDx0ZXh0IHg9IjExNSIgeT0iMzMxIiBm
b250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTUs
MjU1LDAuOSkiIHRleHQtYW5jaG9yPSJtaWRkbGUiPmZldGNoIGxvZ3M8L3RleHQ+CiAgPHRleHQg
eD0iMTE1IiB5PSIzNDciIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZp
bGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+dGFnczogIDx0
ZXh0IHg9IjExNSIgeT0iMzYzIiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEw
IiBmaWxsPSJyZ2JhKDI1NSwyNTUsMjU1LDAuOSkiIHRleHQtYW5jaG9yPSJtaWRkbGUiPmZldGNo
IGxvZ3M8L3RleHQ+CiAgPHRleHQgeD0iMTE1IiB5PSIzNzkiIGZvbnQtZmFtaWx5PSJtb25vc3Bh
```

xyZWN0IHg9IjMwIiB5PSIxNzAiIHdpZHRoPSI3NDAiIGhlaWdodD0iMTE1IiByeD0iOCIgZmlsbD0
iI2ZmZiIgc3Ryb2tlPSIjZTJlOGYwIiBzdHJva2Utd2lkdGg9IjIiLz4KICA8dGV4dCB4PSI0MDAi
IHk9IjE5NSIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEyIiBmb
250LXdlaWdodD0iYm9sZCIgZmlsbD0iIzMzMyIgdGV4dC1hbmNob3I9Im1pZGRsZSI+U2VnbWVudC
B2cy4gTWFuYWdlbWVudCBab25lIEZpbHRlcmluZzwvdGV4dD4KCiAgPCEtLSBDb21wYXJpc29uIGJ
veGVzIC0tPgogIDxyZWN0IHg9IjUwIiB5PSIyMTAiIHdpZHRoPSIzNDAiIGhlaWdodD0iNjAiIHJ4
PSI2IiBmaWxsPSIjZmVjYWNhIi8+CiAgPHRleHQgeD0iMjIwIiB5PSIyMzIiIGZvbnQtZmFtaWx5P
SJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZm9udC13ZWlnaHQ9ImJvbGQiIGZpbG
w9IiNkYzI2MjYiIHRleHQtYW5jaG9yPSJtaWRkbGUiPk1hbmFnZW1lbnQgWm9uZXMgKExlZ2FjeSk
8L3RleHQ+CiAgPHRleHQgeD0iMjIwIiB5PSIyNTAiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1z
ZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzdmMWQxZCIgdGV4dC1hbmNob3I9Im1pZGRsZSI+U
HJlY2FsY3VsYXRlZCDigKIgUGVyZm9ybWFuY2UgYm90dGxlbmVjayDigKIgTm8gR3JhaWwg4oCiIF
NpbmdsZSBkaW1lbnNpb248L3RleHQ+CgogIDxyZWN0IHg9IjQxMCIgeT0iMjEwIiB3aWR0aD0iMzQ
wIiBoZWlnaHQ9IjYwIiByeD0iNiIgZmlsbD0iI2QxZmFlNSIvPgogIDx0ZXh0IHg9IjU4MCIgeT0i
MjMyIiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZvbnQtd
2VpZ2h0PSJib2xkIiBmaWxsPSIjMDU5NjY5IiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5TZWdtZW50cy
AoTW9kZXJuKTwvdGV4dD4KICA8dGV4dCB4PSI1ODAiIHk9IjI1MCIgZm9udC1mYW1pbHk9IkFyaWF
sLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjMDY0ZTNiIiB0ZXh0LWFuY2hvcj0i
bWlkZGxlIj5RdWVyeS10aW1lIOKAoiBIaWdobHkgc2NhbGFibGUg4oCiIEZ1bGwgR3JhaWwg4oCiI
E11bHRpLWRpbWVuc2lvbmFsICsgVmFyaWFibGVzPC90ZXh0Pgo8L3N2Zz4K)

````
When a segment is active:
1. User executes a DQL query or opens an app
2. Segment conditions are injected into the query
3. Grail evaluates only conditions relevant to the data type
4. Filtered results returned

```
User Query:           fetch logs | limit 100
Active Segment:       filter matchesValue(tags, "env:production")
Effective Query:      fetch logs | filter matchesValue(tags,
"env:production") | limit 100
```

### Segment Condition Evaluation

- `fetch logs` → Only log-related conditions apply
- `fetch spans` → Only span-related conditions apply
- `timeseries` → Only metric-related conditions apply

### Multiple Conditions (OR Logic)

Multiple conditions for the same data type in a segment are OR-combined:

```dql
// Segment with multiple conditions
````
````

```
filter dt.entity.service == "SERVICE-123"
filter dt.entity.service == "SERVICE-456"

// Results in: Show data for SERVICE-123 OR SERVICE-456
```

---

3. Creating Segments

Via Segments App

1. Navigate to **Segments** app in Dynatrace
2. Click **Create segment**
3. Configure:
   - **Name**: Descriptive segment name
   - **Description**: Purpose and scope
   - **Filter conditions**: DQL filter expressions
4. Click **Save**

Segment Filter Syntax

Segments use DQL filter syntax:

```dql
// Basic entity filter
filter dt.entity.host == "HOST-ABC123"

// Tag-based filter
filter matchesValue(tags, "env:production")

// Multiple conditions (AND)
filter matchesValue(tags, "env:production") and matchesValue(tags,
"team:frontend")

// Pattern matching
filter matchesPhrase(dt.entity.service.name, "payment")
```

---

4. Segment Examples

Example 1: Team-Based Segment

Replaces: Management Zone filtering by team ownership

Segment Name: `Frontend Team`

Description: Shows data for frontend team services
Filter:
```dql
filter matchesValue(tags, "team:frontend")
```

Example 2: Environment Segment

Replaces: Production Management Zone

Segment Name: `Production Environment`
Description: Filters to production environment only
Filter:
```dql
filter matchesValue(tags, "env:production")
```

Example 3: Regional Segment

Replaces: Geographic Management Zone

Segment Name: `North America`
Description: Data from NA region infrastructure
Filter:
```dql
filter matchesValue(tags, "region:us-east-1") or matchesValue(tags,
"region:us-west-2")
```

Example 4: Application Segment

Replaces: Application-specific Management Zone

Segment Name: `Payment System`
Description: Payment system services and dependencies
Filter:
```dql
filter matchesPhrase(dt.entity.service.name, "payment")
   or matchesPhrase(dt.entity.service.name, "checkout")
```

---

5. Variables in Segments

What Are Variables?

Variables make segments dynamic - users can select values at runtime instead

````
of hardcoding them.

### Variable Types

| Type | Description | Example |
|------|-------------|---------|
| **Entity** | Select from entities | Kubernetes clusters |
| **String** | Free text input | Environment name |
| **List** | Predefined options | [prod, staging, dev] |

### Creating a Variable Segment

**Segment Name**: `Dynamic Environment`
**Variable**: `$environment` (type: list, values: production, staging,
development)
**Filter**:
```dql
filter matchesValue(tags, concat("env:", $environment))
```

### Entity Variable Example

**Segment Name**: `By Kubernetes Cluster`
**Variable**: `$cluster` (type: entity, entity type: kubernetes_cluster)
**Filter**:
```dql
filter dt.entity.kubernetes_cluster == $cluster
```

Users can then select which cluster to filter by when using the segment.

---

## 6. Testing Segments with DQL

### Validate Segment Logic

Before creating a segment, test the filter logic with DQL queries:

```dql
// Test: Team-based filter for segment
// Verify this returns expected services
fetch dt.entity.service
| filter matchesValue(tags, "team:frontend")
| fields entity.name, tags
| limit 20
```
````

```dql
// Test: Environment filter for segment
// Should return only production entities
fetch dt.entity.host
| filter matchesValue(tags, "env:production")
| fields entity.name, tags
| limit 20
```

```dql
// Test: Combined filter (environment AND team)
// For layered segment filtering
fetch dt.entity.service
| filter matchesValue(tags, "env:production")
    and matchesValue(tags, "team:frontend")
| fields entity.name, tags
| limit 20
```

Test Segment on Logs

```dql
// Test log filtering with segment condition
// Validates segment will work for log queries
fetch logs
| filter matchesValue(dt.entity.service.tags, "team:frontend")
| fields timestamp, content, dt.entity.service
| sort timestamp desc
| limit 20
```

---

7. Mapping MZ Rules to Segment Filters

Common MZ Rule Patterns

MZ Rule Type	Segment Filter Equivalent
Host tag equals	`filter matchesValue(tags, "key:value")`
Service name contains	`filter matchesPhrase(dt.entity.service.name, "text")`
Process group tag	`filter matchesValue(dt.entity.process_group.tags, "key:value")`
Kubernetes namespace	`filter dt.entity.cloud_application.namespace == "namespace"`
Cloud provider tag	`filter matchesValue(tags, "cloud:aws")`

````
### Conversion Examples

**MZ Rule**: Host with tag `env` equals `production`
**Segment Filter**:
```dql
filter matchesValue(tags, "env:production")
```

**MZ Rule**: Service name contains `payment`
**Segment Filter**:
```dql
filter matchesPhrase(dt.entity.service.name, "payment")
```

**MZ Rule**: Host in AWS
**Segment Filter**:
```dql
filter matchesValue(tags, "cloud:aws")
```

**MZ Rule**: Kubernetes namespace equals `production`
**Segment Filter**:
```dql
filter dt.entity.cloud_application.namespace == "production"
```

---

## 8. Using Segments in Apps

### Segment Selection

Most Grail-based apps have a segment selector:
- **Location**: Usually top of the app interface
- **Multiple selection**: Can select multiple segments (layered)
- **Persistence**: Selection persists across navigation

### Dashboard Integration

Segments can be applied at two levels:

1. **Dashboard level**: Applies to all tiles
2. **Tile level**: Overrides dashboard segment for specific tile

### Apps Supporting Segments

| App | Segment Support |
|-----|----------------|
````

```
| Logs | ✅ Full support |
| Distributed Traces | ✅ Full support |
| Services | ✅ Full support |
| Dashboards | ✅ Dashboard + Tile level |
| Notebooks | ✅ Full support |
| Problems | ✅ Full support |

### Classic Apps

Classic apps still use Management Zones. During migration:
- Use MZs for classic apps
- Use Segments for new apps
- Both can coexist during transition

---

## 9. Segment Sharing and Permissions

### Sharing Segments

Segments can be:
- **Private**: Only creator can see/use
- **Shared**: Available to specified users/groups
- **Public**: Available to all users in environment

### Required Permissions

| Action | Required Permission |
|--------|---------------------|
| View segments | `storage:filter-segments:read` |
| Create segments | `storage:filter-segments:write` |
| Share segments | `storage:filter-segments:share` |
| Delete segments | `storage:filter-segments:delete` |

These permissions are included in default policies:
- **Dynatrace Standard User**: Read, Write, Share
- **Dynatrace Professional User**: Read, Write, Share, Delete

---

## 10. Segment Best Practices

### Design Principles

1. **Align with business structure**: Teams, products, regions
2. **Use consistent naming**: Clear, descriptive names
3. **Leverage variables**: For flexibility
4. **Test thoroughly**: Verify filtering works as expected
```

5. **Document purpose**: Help users understand when to use

Naming Conventions

```
// Good segment names
"Production Environment"
"Frontend Team Services"
"North America Region"
"Payment System"

// Bad segment names
"Segment 1"
"Test"
"My filter"
```

Performance Considerations

- Segments are evaluated at query time
- Complex filters may impact performance
- Use indexed fields when possible (tags, entity IDs)
- Avoid expensive operations like regex on large datasets

Migration Checklist

- [ ] Identify all MZ filtering use cases
- [ ] Design segment for each use case
- [ ] Test segment filter logic with DQL
- [ ] Create segment in Segments app
- [ ] Share segment with appropriate users
- [ ] Update dashboards to use segments
- [ ] Train users on segment selection

Summary

In this notebook, you learned:

1. **Segment fundamentals**: DQL-powered, query-time filters
2. **Creating segments**: Via Segments app with DQL filters
3. **Variables**: Making segments dynamic with runtime values
4. **MZ mapping**: Converting MZ rules to segment filters
5. **App integration**: How segments work across Dynatrace apps

Next Steps

```
Continue to **MZ2POL-06: Migration Execution** to:
- Execute the migration plan step-by-step
- Handle parallel running period
- Perform cutover from MZs to new model

## Additional Resources

- [Segments in DQL Queries]
(https://docs.dynatrace.com/docs/manage/segments/concepts/segments-concepts-queries)
- [Log Filtering with Segments](https://www.dynatrace.com/news/blog/log-filtering-made-easy-data-segmentation-and-advanced-filters-in-dynatrace-logs/)
- [Power Dashboarding: Filter Data Effectively]
(https://www.dynatrace.com/news/blog/power-dashboarding-3-filter-data-effectively/)
```