```
# OPMIG-07: Metric & Event Extraction

> **Series:** OPMIG | **Notebook:** 7 of 9 | **Created:** December 2025

> **OpenPipeline Migration Series** | Notebook 7 of 9
> **Level:** Intermediate to Advanced
> **Estimated Time:** 70 minutes

---

## Learning Objectives

By completing this notebook, you will:

1. Configure metric and event extraction from logs
2. Generate business events for analytics
3. ⭐ **NEW:** Implement SLI metrics using RED methodology (Rate, Errors,
Duration)
4. ⭐ **NEW:** Extract business KPIs (revenue, conversion, user journey)
5. ⭐ **NEW:** Apply metric design best practices (cardinality, naming,
aggregation)
6. Validate extractions with DQL queries
7. Optimize extraction for cost efficiency

---



---

## Extraction Stage Overview

The Extraction stage runs **after** Processing and **before** Storage. It
generates:

![Extraction Stage]
(
ciIHZpZXdCb3g9IjAgMCA4MDAgMzAwIj4KICA8ZGVmcz4KICAgIDxsaW5lYXJHcmFkaWVudCBpZD0
ibWV0cmljR3JhZCIgeDE9IjAlIiB5MT0iMCUiIHgyPSIxMDAlIiB5Mj0iMTAwJSI+CiAgICAgIDxz
dG9wIG9mZnNldD0iMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiMzYjgyZjY7c3RvcC1vcGFjaXR5OjEiI
C8+CiAgICAgIDxzdG9wIG9mZnNldD0iMTAwJSIgc3R5bGU9InN0b3AtY29sb3I6IzI1NjNlYjtzdG
9wLW9wYWNpdHk6MSIgLz4KICAgIDwvbGluZWFyR3JhZGllbnQ+CiAgICA8bGluZWFyR3JhZGllbnQ
gaWQ9ImV2ZW50R3JhZCIgeDE9IjAlIiB5MT0iMCUiIHgyPSIxMDAlIiB5Mj0iMTAwJSI+CiAgICAg
IDxzdG9wIG9mZnNldD0iMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiNmNTllMGI7c3RvcC1vcGFjaXR5O
jEiIC8+CiAgICAgIDxzdG9wIG9mZnNldD0iMTAwJSIgc3R5bGU9InN0b3AtY29sb3I6I2Q5NzcwNj
tzdG9wLW9wYWNpdHk6MSIgLz4KICAgIDwvbGluZWFyR3JhZGllbnQ+CiAgICA8bGluZWFyR3JhZGl
lbnQgaWQ9ImJpekdyYWQiIHgxPSIwJSIgeTE9IjAlIiB4Mj0iMTAwJSIgeTI9IjEwMCUiPgogICAg
ICA8c3RvcCBvZmZzZXQ9IjAlIiBzdHlsZT0ic3RvcC1jb2xvcjojMTBiOTgxO3N0b3Atb3BhY2l0e

```

ToxIiAvPgogICAgICA8c3RvcCBvZmZzZXQ9IjEwMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiMwNTk2Nj
k7c3RvcC1vcGFjaXR5OjEiIC8+CiAgICA8L2xpbmVhckdyYWRpZW50PgogICAgPGZpbHRlciBpZD0
iZXhTaGFkb3ciPgogICAgICA8ZmVEcm9wU2hhZG93IGR4PSIxIiBkeT0iMSIgc3RkRGV2aWF0aW9u
PSIyIiBmbG9vZC1vcGFjaXR5PSIwLjE1Ii8+CiAgICA8L2ZpbHRlcj4KICAgIDxtYXJrZXIgaWQ9I
mV4QXJyb3ciIG1hcmtlcldpZHRoPSIxMCIgbWFya2VySGVpZ2h0PSI3IiByZWZYPSI5IiByZWZZPS
IzLjUiIG9yaWVudD0iYXV0byI+CiAgICAgIDxwb2x5Z29uIHBvaW50cz0iMCAwLCAxMCAzLjUsIDA
gNyIgZmlsbD0iIzY0NzQ4YiIvPgogICAgPC9tYXJrZXI+CiAgPC9kZWZzPgoKICA8IS0tIEJhY2tn
cm91bmQgLS0+CiAgPHJlY3Qgd2lkdGg9IjgwMCIgaGVpZ2h0PSIzMDAiIGZpbGw9IiNmOGY5ZmEiI
HJ4PSIxMCIvPgoKICA8IS0tIFRpdGxlIC0tPgogIDx0ZXh0IHg9IjQwMCIgeT0iMjgiIGZvbnQtZm
FtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxOCIgZm9udC13ZWlnaHQ9ImJvbGQ
iIGZpbGw9IiMzMzMiIHRleHQtYW5jaG9yPSJtaWRkbGUiPkV4dHJhY3Rpb24gU3RhZ2U8L3RleHQ+
CiAgPHRleHQgeD0iNDAwIiB5PSI0OCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb
250LXNpemU9IjExIiBmaWxsPSIjNjY2IiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5HZW5lcmF0ZSBkZX
JpdmVkIHNpZ25hbHMgZnJvbSBsb2dzIGJlZm9yZSBzdG9yYWdlPC90ZXh0PgoKICA8IS0tIElucHV
0IC0tPgogIDxyZWN0IHg9IjMwIiB5PSI5MCIgd2lkdGg9IjE0MCIgaGVpZ2h0PSIxMjAiIHJ4PSI4
IiBmaWxsPSIjZmZmIiBzdHJva2U9IiNlMmU4ZjAiIHN0cm9rZS13aWR0aD0iMiIgZmlsdGVyPSJ1c
mwoI2V4U2hhZG93KSIvPgogIDx0ZXh0IHg9IjEwMCIgeT0iMTIwIiBmb250LWZhbWlseT0iQXJpYW
wsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTEiIGZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSIjMzM
zIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5Qcm9jZXNzZWQgTG9nPC90ZXh0PgogIDx0ZXh0IHg9IjEw
MCIgeT0iMTQ1IiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSIjN
jY2IiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5sb2dsZXZlbDogRVJST1I8L3RleHQ+CiAgPHRleHQgeD
0iMTAwIiB5PSIxNTgiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw
9IiM2NjYiIHRleHQtYW5jaG9yPSJtaWRkbGUiPmR1cmF0aW9uX21zOiAxNTA8L3RleHQ+CiAgPHRl
eHQgeD0iMTAwIiB5PSIxNzEiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiI
GZpbGw9IiM2NjYiIHRleHQtYW5jaG9yPSJtaWRkbGUiPm9yZGVyX2lkOiAxMjM0NTwvdGV4dD4KIC
A8dGV4dCB4PSIxMDAiIHk9IjE4NCIgZm9udC1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1zaXplPSI
xMCIgZmlsbD0iIzY2NiIgdGV4dC1hbmNob3I9Im1pZGRsZSI+YW1vdW50OiA5OS45OTwvdGV4dD4K
ICA8dGV4dCB4PSIxMDAiIHk9IjE5NyIgZm9udC1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1zaXplP
SIxMCIgZmlsbD0iIzY2NiIgdGV4dC1hbmNob3I9Im1pZGRsZSI+c2VydmljZS5uYW1lOiBwYXk8L3
RleHQ+CgogIDwhLS0gQXJyb3dzIHRvIGV4dHJhY3Rpb25zIC0tPgogIDxwYXRoIGQ9Ik0xNzAsMTI
wIEwyMzAsOTAiIHN0cm9rZT0iIzNiODJmNiIgc3Ryb2tlLXdpZHRoPSIyIiBmaWxsPSJub25lIiBt
YXJrZXItZW5kPSJ1cmwoI2V4QXJyb3cpIi8+CiAgPHBhdGggZD0iTTE3MCwxNTAgTDIzMCwxNTAiI
HN0cm9rZT0iI2Y1OWUwYiIgc3Ryb2tlLXdpZHRoPSIyIiBmaWxsPSJub25lIiBtYXJrZXItZW5kPS
J1cmwoI2V4QXJyb3cpIi8+CiAgPHBhdGggZD0iTTE3MCwxODAgTDIzMCwyMTAiIHN0cm9rZT0iIzE
wYjk4MSIgc3Ryb2tlLXdpZHRoPSIyIiBmaWxsPSJub25lIiBtYXJrZXItZW5kPSJ1cmwoI2V4QXJy
b3cpIi8+CgogIDwhLS0gTWV0cmljIEV4dHJhY3Rpb24gLS0+CiAgPHJlY3QgeD0iMjQwIiB5PSI2N
SIgd2lkdGg9IjIzMCIgaGVpZ2h0PSI1NSIgcng9IjgiIGZpbGw9InVybCgjbWV0cmljR3JhZCkiIG
ZpbHRlcj0idXJsKCNleFNoYWRvdykiLz4KICA8dGV4dCB4PSIzNTUiIHk9Ijg4IiBmb250LWZhbWl
seT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTEiIGZvbnQtd2VpZ2h0PSJib2xkIiBm
aWxsPSJ3aGl0ZSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+TWV0cmljIEV4dHJhY3Rpb248L3RleHQ+C
iAgPHRleHQgeD0iMzU1IiB5PSIxMDUiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT
0iMTAiIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+bG9
nLmFwaS5kdXJhdGlvbl9tcyA9IDE1MDwvdGV4dD4KICA8dGV4dCB4PSIzNTUiIHk9IjExNSIgZm9u
dC1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1zaXplPSIxMCIgZmlsbD0icmdiYSgyNTUsMjU1LDI1N
SwwLjgpIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5kaW1zOiBzZXJ2aWNlLm5hbWUsIGxvZ2xldmVsPC
90ZXh0PgoKICA8IS0tIEV2ZW50IEV4dHJhY3Rpb24gLS0+CiAgPHJlY3QgeD0iMjQwIiB5PSIxMzA
iIHdpZHRoPSIyMzAiIGhlaWdodD0iNTUiIHJ4PSI4IiBmaWxsPSJ1cmwoI2V2ZW50R3JhZCkiIGZp
bHRlcj0idXJsKCNleFNoYWRvdykiLz4KICA8dGV4dCB4PSIzNTUiIHk9IjE1MyIgZm9udC1mYW1pb

```
Hk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjExIiBmb250LXdlaWdodD0iYm9sZCIgZm
lsbD0id2hpdGUiIHRleHQtYW5jaG9yPSJtaWRkbGUiPkV2ZW50IEV4dHJhY3Rpb248L3RleHQ+CiA
gPHRleHQgeD0iMzU1IiB5PSIxNzAiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0i
MTAiIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+cGF5b
WVudC5lcnJvcjogT3JkZXIgMTIzNDUgZmFpbGVkPC90ZXh0PgogIDx0ZXh0IHg9IjM1NSIgeT0iMT
gwIiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSJyZ2JhKDI1NSw
yNTUsMjU1LDAuOCkiIHRleHQtYW5jaG9yPSJtaWRkbGUiPnR5cGU6IEVSUk9SIOKGkiBEYXZpcyBB
STwvdGV4dD4KICAgPCEtLSBCdXNpbmVzcyBFdmVudCBFeHRyYWN0aW9uIC0tPgogIDxyZWN0IHg9I
jI0MCIgeT0iMTk1IiB3aWR0aD0iMjMwIiBoZWlnaHQ9IjU1IiByeD0iOCIgZmlsbD0idXJsKCNiaX
pHcmFkKSIgZmlsdGVyPSJ1cmwoI2V4U2hhZG93KSIvPgogIDx0ZXh0IHg9IjM1NSIgeT0iMjE4IiB
mb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTEiIGZvbnQtd2VpZ2h0
PSJib2xkIiBmaWxsPSJ3aGl0ZSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+QnVzaW5lc3MgRXZlbnRzI
Xh0cmFjdGlvbjwvdGV4dD4KICA8dGV4dCB4PSIzNTUiIHk9IjIzNSIgZm9udC1mYW1pbHk9Im1vbm
9zcGFjZSIgZm9udC1zaXplPSIxMCIgZmlsbD0icmdiYSgyNTUsMjU1LDI1NSwwLjkpIiB0ZXh0LWF
uY2hvcj0ibWlkZGxlIj5vcmRlci5vcm9jZXNzZWQ6ICQ5OS45OTwvdGV4dD4KICA8dGV4dCB4PSIz
NTUiIHk9IjI0NSIgZm9udC1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1zaXplPSIxMCIgZmlsbD0ic
mdiYSgyNTUsMjU1LDI1NSwwLjgpIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj7ihpIgR3JhaWwgYml6ZX
ZlbnRzIHRhYmxlPC90ZXh0PgoKICA8IS0tIEFycm93cyB0byBvdXRwdXRzIC0tPgogIDxwYXRoIGQ
9Ik00NzAsOTAgTDUyMCw5MCIgc3Ryb2tlPSIjM2I4MmY2IiBzdHJva2Utd2lkdGg9IjIiIGZpbGw9
Im5vbmUiIG1hcmtlci1lbmQ9InVybCgjZXhBcnJvdykiLz4KICA8cGF0aCBkPSJNNDcwLDE1NyBMN
TIwLDE1NyIgc3Ryb2tlPSIjZjU5ZTBiIiBzdHJva2Utd2lkdGg9IjIiIGZpbGw9Im5vbmUiIG1hcm
tlci1lbmQ9InVybCgjZXhBcnJvdykiLz4KICA8cGF0aCBkPSJNNDcwLDIyMiBMNTIwLDIyMiIgc3R
yb2tlPSJMTBiOTgxIiBzdHJva2Utd2lkdGg9IjIiIGZpbGw9Im5vbmUiIG1hcmtlci1lbmQ9InVy
bCgjZXhBcnJvdykiLz4KICAgPCEtLSBPdXRwdXRzIC0tPgogIDxyZWN0IHg9IjUzMCIgeT0iUzMCIgeT0iNjUiI
HdpZHRoPSIyNDAiIGhlaWdodD0iNTUiIHJ4PSI4IiBmaWxsPSIjZGJlYWZlIiBzdHJva2U9IiMzYj
gyZjYiIHN0cm9rZS13aWR0aD0iMSIvPgogIDx0ZXh0IHg9IjY1MCIgeT0iODUiIGZvbnQtZmFtaWx
5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZm9udC13ZWlnaHQ9ImJvbGQiIGZp
bGw9IiMxZTQwYWYiIHRleHQtYW5jaG9yPSJtaWRkbGUiPk1ldHJpY3MgKDEwIHllYXIgcmV0ZW50a
W9uKTwvdGV4dD4KICA8dGV4dCB4PSI2NTAiIHk9IjEwMCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW
5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTUsMjU1LDAuNykiIHRleH
QtYW5jaG9yPSJtaWRkbGUiPnU1MGFmIiB0ZXh0LWFuY2hvcj0ibWlkZGx
lIj5TTEkgZGFzaGJvYXJkcywgYWxlcnRzPC90ZXh0PgogIDx0ZXh0IHg9IjY1MCIgeT0iMTEyIiBm
b250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9InJnYmEo
WYiIHRleHQtYW5jaG9yPSJtaWRkbGUiPkxvdyBjYXJkaW5hbGl0eSByZXF1aXJlZDwvdGV4dD4KICA
AgPHJlY3QgeD0iNTMwIiB5PSIxMzAiIHdpZHRoPSIyNDAiIGhlaWdodD0iNTUiIHJ4PSI4IiBmaWx
sPSIjZmVmM2M3IiBzdHJva2U9IiNmNTllMGIiIHN0cm9rZS13aWR0aD0iMSIvPgogIDx0ZXh0IHg9
IjY1MCIgeT0iMTUwIiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iM
TAiIGZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSIjOTI0MDBlIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj
5FdmVudHMgKERhdmlzIEFJKTwvdGV4dD4KICA8dGV4dCB4PSI2NTAiIHk9IjE2NSIgZm9udC1mYW1
pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjOTI0MDBlIiB0ZXh0
LWFuY2hvcj0ibWlkZGxlIj5Sb290IGNhdXNlIGFuYWx5c2lzPC90ZXh0PgogIDx0ZXh0IHg9IjY1M
CIgeT0iMTc3IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIG
ZpbGw9IiM5MjQwMGUiIHRleHQtYW5jaG9yPSJtaWRkbGUiPkF1dG9tYXRpYyBwcm9ibGVtIGRldGV
jdGlvbjwvdGV4dD4KICAgPCEtLSBCdXNpbmVzcyBFdmVudCBFeHRyYWN0aW9uIC0tPgogIDxyZWN0
NTUiIHk9IjI0MSIgeD0iMzAiIHdpZHRoPSIyNDAiIGhlaWdodD0iNTUiIHJ4PSI4IiBmaWxsPSIjZ
DFmYWU1IiBzdHJva2U9IiMxMGI5ODEiIHN0cm9rZS13aWR0aD0iMSIvPgogIDx0ZXh0IHg9IjE1M
SIvPgogIDx0ZXh0IHg9IjE1MCIgeT0iMjE1IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaW
YiIGZvbnQtc2l6ZT0iMTAiIGZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSIjMDQ3ODU3IiB0ZXh0LWF
uY2hvcj0ibWlkZGxlIj5vcmRlci5vcm9jZXNzZWQ6ICQ5CdXNpbmVzcyBFdmVudHMgKEdyYWlsKTwvdGV4dD4KICA8dGV4dCB4PSI2NTAiIHk9IjI
NTAiIHk9IjIxMCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwI
```

iBmaWxsPSIjMDQ3ODU3IiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5SZXZlbnVlIHRyYWNraW5nLCBmdW
5uZWxzPC90ZXh0PgogIDx0ZXh0IHg9IjY1MCIgeT0iMjQyIiBmb250LWZhbWlseT0iQXJpYWwsIHN
hbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiMwNDc4NTciIHRleHQtYW5jaG9yPSJtaWRk
bGUiPkN1c3RvbSBhbmFseXRpY3M8L3RleHQ+CgogIDwhLS0gIDwhLS0gIDxyZWN0IHg9IjMwIi
iB5PSIyNjAiIHdpZHRoPSI3NDAiIGhlaWdodD0iMzAiIHJ4PSI0IiBmaWxsPSIjZTBmMmZlIi8+Ci
AgPHRleHQgeD0iNDAwIiB5PSIyODAiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9
udC1zaXplPSIxMCIgZmlsbD0iIzAzNjlhMSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+UHJvIFRpcDog
RXh0cmFjdCBtZXRyaWNzIGZyb20gaGlnaC12b2x1bWUgbG9ncywgdGhlbiBEUk9QIHRoZSBsb2dz
HRvIHNhdmUgOTklKyBvbiBzdG9yYWdlPC90ZXh0Pgo8L3N2Zz4K)

Key Benefits

Benefit	Description
Derived signals	Create metrics from logs without app changes
Cost optimization	Extract metrics, then drop verbose logs
Davis AI integration	Generated events feed into AI analysis
Business visibility	Business events enable analytics dashboards

---

Metric Extraction

OpenPipeline supports two types of metrics:

Value Metrics (Gauge)

Extract numeric values as gauge metrics.

Field	Description
Key	Metric name (e.g., `log.payment.amount`)
Value	Numeric field to extract
Dimensions	Fields to use as metric dimensions
Matching Condition	When to apply extraction

Example: Extract payment amount
```
Metric Type: Value
Key: log.payment.amount
Value: amount
Dimensions: service.name, environment, status
Matching: contains(content, "payment") AND isNotNull(amount)
```

Counter Metrics

````
Count occurrences of matching records.

| Field | Description |
|-------|-------------|
| **Key** | Metric name (e.g., `log.error.count`) |
| **Dimensions** | Fields to use as metric dimensions |
| **Matching Condition** | When to count |

**Example: Count errors**
```
Metric Type: Counter
Key: log.errors.count
Dimensions: service.name, error_code
Matching: loglevel == "ERROR"
```

### Metric Naming Conventions

```
Format: ..

Examples:
  log.api.request_count
  log.payment.amount
  log.auth.failed_attempts
  span.checkout.duration_ms
```

### Dimension Best Practices

| Do | Don't |
|----|-------|
| Low-cardinality fields | High-cardinality IDs |
| `service.name` | `request_id` |
| `environment` | `user_id` |
| `error_code` | `timestamp` |

---

## SLI Metric Patterns: RED Methodology ⭐ NEW

The **RED Method** (Rate, Errors, Duration) provides comprehensive service observability.

### RED Overview

| Metric | Measures | Why |
````

````
|--------|----------|-----|
| **Rate** | Requests/sec | Traffic volume, capacity |
| **Errors** | Error rate (%) | Reliability, SLA |
| **Duration** | Response time (p50, p95, p99) | User experience |

### Rate: Request Throughput

```
Type: Counter
Key: log.api.request_rate
Dimensions: method, path, status_category, service.name
Matching: isNotNull(path)
```

### Errors: Error Rate SLI

**Total Requests:**
```
Counter: log.api.request_total
Dimensions: service.name, endpoint
```

**Error Requests:**
```
Counter: log.api.request_errors
Dimensions: service.name, endpoint, error_type
Matching: status >= 500 OR loglevel == "ERROR"
```

### Duration: Latency Percentiles

```
Type: Value
Key: log.api.response_time_ms
Value: duration_ms
Dimensions: service.name, endpoint, status_category
```

### Latency Buckets (Apdex-style SLI)

Use `fieldsAdd` to create latency buckets:
```
fieldsAdd latency_bucket = if(duration_ms < 100, "fast",
                           else: if(duration_ms < 500, "acceptable",
                           else: if(duration_ms < 1000, "slow",
                           else: "very_slow")))
```
````

````
Then create a counter metric:
```
Counter: log.api.latency_bucket_count
Dimensions: latency_bucket, service.name, endpoint
```

---

---

## Event Extraction

Extract events that Davis AI can analyze for root cause analysis.

### Event Configuration

| Field | Description |
|-------|-------------|
| **Event Name** | Unique event type identifier |
| **Event Description** | Template for event description |
| **Matching Condition** | When to generate event |
| **Event Type** | Category: INFO, AVAILABILITY, ERROR, RESOURCE, CUSTOM |

### Event Types

| Type | Use Case |
|------|----------|
| `INFO` | Informational events |
| `AVAILABILITY` | Service up/down events |
| `ERROR` | Error conditions |
| `RESOURCE` | Resource contention events |
| `CUSTOM` | Custom event types |

### Event Description Templates

Use field placeholders in descriptions:

```
Payment failed for order {order_id}: {error_message}
Service {service.name} error rate exceeded threshold
Authentication failure for user {user_id} from IP {client_ip}
```

### Example Event Extraction

```
Event Name: payment.failure
Event Description: Payment failed for order {order_id}: {error_message}
````
````

````
Event Type: ERROR
Matching: contains(content, "payment") AND contains(content, "failed")
```

---

## Business Event Extraction

Business events are stored in Grail for business analytics.

### Business Event Configuration

| Field | Description |
|-------|-------------|
| **Event Type** | Business event type name |
| **Event Provider** | Source system identifier |
| **Data Fields** | Fields to include in event |
| **Matching Condition** | When to extract |

### Example: Order Events

```
Event Type: com.example.order.placed
Event Provider: checkout-service
Data Fields:
  - order_id
  - customer_id
  - total_amount
  - currency
  - timestamp
Matching: contains(content, "order placed successfully")
```

### Querying Business Events

---

## Business KPI Extraction ⭐ NEW

### Revenue Tracking

**Order Revenue:**
```
Type: Value
Key: log.revenue.order_amount
Value: amount
Dimensions: currency, service.name, payment_method
```
````
````

````
### Conversion Funnels

**E-Commerce Funnel:**
1. Product View → 2. Add to Cart → 3. Checkout → 4. Complete

Use `fieldsAdd` to categorize funnel stages:
```
fieldsAdd funnel_stage = if(action == "product_view", "1_view",
                         else: if(action == "add_to_cart", "2_cart",
                         else: if(action == "checkout_started", "3_checkout",
                         else: "4_complete")))
```

### Feature Usage

```
Counter: log.feature.usage_count
Dimensions: feature_name, feature_enabled, user_segment
```

---

```python
// Query business events extracted from logs
fetch bizevents, from: now() - 24h
| filter event.type == "com.example.order.placed"
| fields timestamp, order_id, customer_id, total_amount
| limit 50
```

```python
// Business event analytics - order totals by hour
fetch bizevents, from: now() - 24h
| filter event.type == "com.example.order.placed"
| makeTimeseries {
    order_count = count(),
    total_revenue = sum(total_amount)
  }, interval: 1h
```

---

## Extraction Matching Conditions

Matching conditions determine which records trigger extraction.

### Common Patterns
````

````
| Pattern | Condition |
|---------|-----------|
| Error logs | `loglevel == "ERROR"` |
| Specific service | `service.name == "payment-service"` |
| Contains text | `contains(content, "failed")` |
| Has field | `isNotNull(order_id)` |
| Numeric threshold | `duration_ms > 1000` |

### Combining Conditions

```
// AND - both must match
loglevel == "ERROR" AND service.name == "payment"

// OR - either matches
status == "failed" OR status == "error"

// Complex conditions
(loglevel == "ERROR" OR loglevel == "WARN")
  AND contains(content, "payment")
  AND isNotNull(amount)
```

### Field Reference in Conditions

Use any field available after processing:
- Original fields from ingestion
- Parsed fields from processors
- Computed fields from `fieldsAdd`

---

## Practical Examples

### Example 1: API Request Metrics

**Source Log:**
```
2024-12-12T10:30:45 INFO - GET /api/users completed in 150ms, status=200
```

**Pipeline Processing:**
```dql
// First, parse the log
parse content, "LD:method ' ' LD:path ' completed in ' INT:duration_ms 'ms,
status=' INT:status_code"
```
````

Metric Extraction 1: Request duration
```
Type: Value
Key: log.api.request_duration_ms
Value: duration_ms
Dimensions: method, path, status_code
Matching: isNotNull(duration_ms)
```

Metric Extraction 2: Request count
```
Type: Counter
Key: log.api.request_count
Dimensions: method, path, status_code
Matching: isNotNull(path)
```

Example 2: Error Event Generation

Source Log:
```
2024-12-12T10:30:45 ERROR PaymentService - Transaction failed: insufficient funds, orderId=12345
```

Event Extraction:
```
Event Name: payment.transaction.failed
Event Description: Transaction failed for order {order_id}: {error_reason}
Event Type: ERROR
Matching: loglevel == "ERROR" AND contains(content, "Transaction failed")
```

Example 3: Login Business Events

Source Log:
```
2024-12-12T10:30:45 INFO AuthService - User login successful: userId=john123, ip=192.168.1.100
```

Business Event Extraction:
```
Event Type: com.example.auth.login
Event Provider: auth-service
Data Fields: user_id, client_ip, timestamp
Matching: contains(content, "login successful")
```

```

### Example 4: Combined Extraction

From a single payment log line, extract:

1. **Metric**: `log.payment.amount` (value: amount)
2. **Metric**: `log.payment.count` (counter)
3. **Event**: Payment processed (INFO)
4. **Business Event**: Order transaction

---

## Validating Extractions

After configuring extractions, verify they're working.

```python
// List log-extracted metrics
// Note: Use the Dynatrace UI (Observe > Metrics) to browse metrics
// Or use timeseries to query a specific metric:
// timeseries avg_value = avg(log.your_metric_name), from: now() - 24h
```

```python
// Query a specific extracted metric
// Replace 'log.api.request_duration_ms' with your metric key
timeseries {
    avg_duration = avg(log.api.request_duration_ms),
    max_duration = max(log.api.request_duration_ms)
  }, from: now() - 2h, interval: 5m
```

```python
// View extracted counter metric by dimension
// Replace 'log.api.request_count' with your metric key
timeseries {
    requests = sum(log.api.request_count)
  }, from: now() - 2h, interval: 5m
```

```python
// View extracted events
fetch events, from: now() - 24h
| filter isNotNull(event.name)
| summarize {event_count = count()}, by: {event.type, event.name}
| sort event_count desc
```
```

```python
// View specific event details
fetch events, from: now() - 24h
| filter event.name == "payment.transaction.failed"
| fields timestamp, event.name, event.description, event.type
| sort timestamp desc
| limit 25
```

```python
// View business events by type
fetch bizevents, from: now() - 24h
| summarize {bizevent_count = count()}, by: {event.type, event.provider}
| sort bizevent_count desc
```

```python
// Compare source log volume to extracted metrics
// This helps verify extraction is capturing expected data
fetch logs, from: now() - 1h
| filter contains(content, "payment")
| summarize {log_count = count()}
| fieldsAdd data_source = "logs"

// Then compare with:
// timeseries sum(log.payment.count)
```

Metric Design Best Practices ⭐ NEW

Cardinality Management

Cardinality	Examples	Max Values	Use?
Low	environment, region	< 50	✅ Ideal
Medium	endpoint, error_code	50-500	✅ Good
High	session_id, request_id	500-10K	⚠️ Avoid
Very High	user_id, order_id	> 10K	❌ Never

Cardinality Example:
```
✅ GOOD: 5 services × 3 environments × 4 methods = 60 time series
❌ BAD: 5 services × 100K user_ids = 500K time series
```

Reduction Techniques:
```dql
// Bucket high-cardinality values
fieldsAdd duration_bucket = if(duration_ms < 100, "<100ms",
                           else: if(duration_ms < 500, "100-500ms",
                           else: ">500ms"))

// Use segments instead of IDs
fieldsAdd user_segment = if(user_tier == "premium", "premium", else: "free")
```

Naming Conventions

Format: `...`

Examples:
```
✅ log.api.request_count
✅ log.api.response_time_ms
✅ log.payment.order_amount_usd
✅ log.auth.failed_login_count

❌ apiReqCnt (abbreviated, no unit)
❌ response_time (no source, no unit)
❌ count (no context)
```

Cost Optimization

Metrics vs. Logs:
```
Scenario: 1M requests/day

Option 1: Store logs (35 days)
  - 35M log records × 500 bytes = 17.5 GB
  - Cost: ~$140/month

Option 2: Extract metrics + drop logs
  - 10 time series, 10 years retention = ~10 MB
  - Cost: ~$1/month
  - Savings: 99.3% 🎉
```

When to Extract:
- ✅ SLI tracking (RED)
- ✅ Business KPIs
- ✅ High-volume health checks
- ❌ Debugging (keep logs)

```
- ❌ Compliance/audit (keep logs)

---



---

## Best Practices

### Metric Extraction

| Practice | Reason |
|----------|--------|
| Use meaningful metric names | Easier discovery and querying |
| Limit dimensions (≤5–7) | Avoid metric cardinality explosion |
| Use low-cardinality dimensions | IDs cause metric bloat |
| Extract only what you need | Reduces storage costs |
| Test matching conditions | Ensure correct data selection |

### Event Extraction

| Practice | Reason |
|----------|--------|
| Meaningful event names | Easier to find in Davis AI |
| Descriptive templates | Context for troubleshooting |
| Appropriate event types | Correct Davis AI handling |
| Avoid over-extraction | Too many events = noise |

### Business Event Extraction

| Practice | Reason |
|----------|--------|
| Consistent event types | Enables analytics |
| Include all needed fields | Avoids re-processing |
| Use namespaced types | Prevents collisions |
| Document event schema | Helps consumers |

### Cost Optimization Pattern

**Extract → Drop → Save**

1. Extract metrics and events from verbose logs
2. Drop the original verbose logs
3. Keep only derived signals

This pattern reduces storage while preserving observability.

---
```

Complete Extraction Pipeline Example

Pipeline: `checkout-service-logs`

Processing Stage (from OPMIG-06):
```dql
// Parse order details
parse content, "'orderId=' INT:order_id ','"
| parse content, "'amount=' DOUBLE:amount"
| parse content, "'status=' LD:order_status"
| parse content, "'duration=' INT:duration_ms 'ms'"
```

Metric Extraction 1: Order Amount
```
Type: Value
Key: log.checkout.order_amount
Value: amount
Dimensions: order_status, service.name
Matching: isNotNull(amount)
```

Metric Extraction 2: Order Count
```
Type: Counter
Key: log.checkout.order_count
Dimensions: order_status, service.name
Matching: isNotNull(order_id)
```

Metric Extraction 3: Processing Duration
```
Type: Value
Key: log.checkout.processing_duration_ms
Value: duration_ms
Dimensions: order_status, service.name
Matching: isNotNull(duration_ms)
```

Event Extraction: Order Failure
```
Event Name: checkout.order.failed
Event Description: Order {order_id} failed with status {order_status}
Event Type: ERROR
Matching: order_status == "failed" OR order_status == "error"
```

````
**Business Event: Order Completed**
```
Event Type: com.example.checkout.order_completed
Event Provider: checkout-service
Data Fields: order_id, amount, order_status, duration_ms
Matching: order_status == "completed" OR order_status == "success"
```

---

## Next Steps

Now that you can extract metrics and events, continue with:

| Notebook | Focus Area |
|----------|------------|
| **OPMIG-08** | Security, Masking & Compliance |
| **OPMIG-09** | Troubleshooting & Validation |

---

## References

- [OpenPipeline Data Extraction](https://docs.dynatrace.com/docs/discover-dynatrace/platform/openpipeline/concepts/data-extraction)
- [Metric Extraction](https://docs.dynatrace.com/docs/discover-dynatrace/platform/openpipeline/use-cases/metric-extraction)
- [Business Event Extraction](https://docs.dynatrace.com/docs/discover-dynatrace/platform/openpipeline/use-cases/bizevent-extraction)
- [Davis AI Events](https://docs.dynatrace.com/docs/platform/davis-ai/basics/events)

---

*Last Updated: December 12, 2025*
````