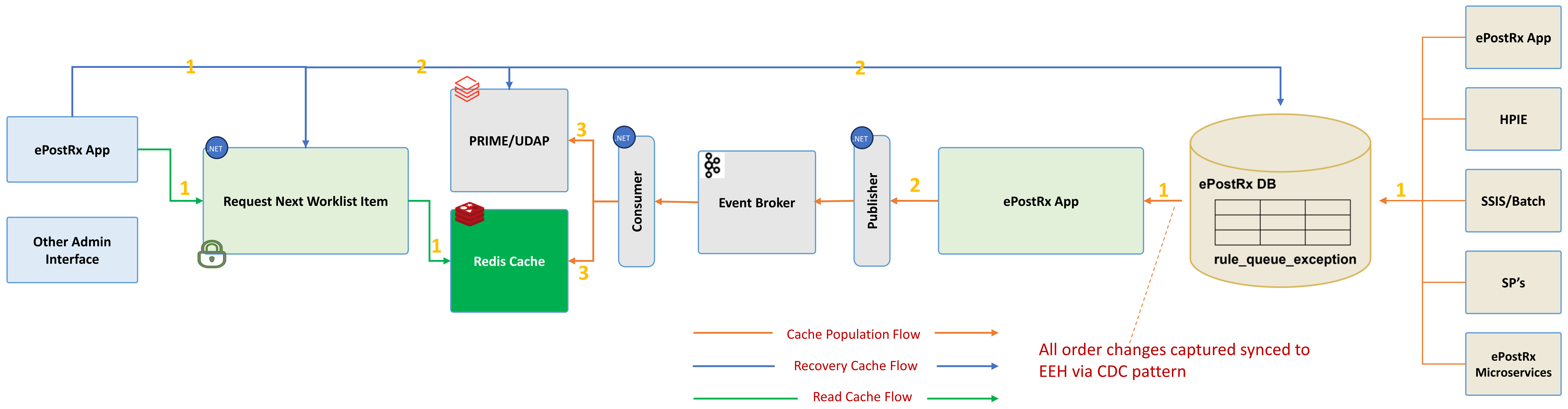
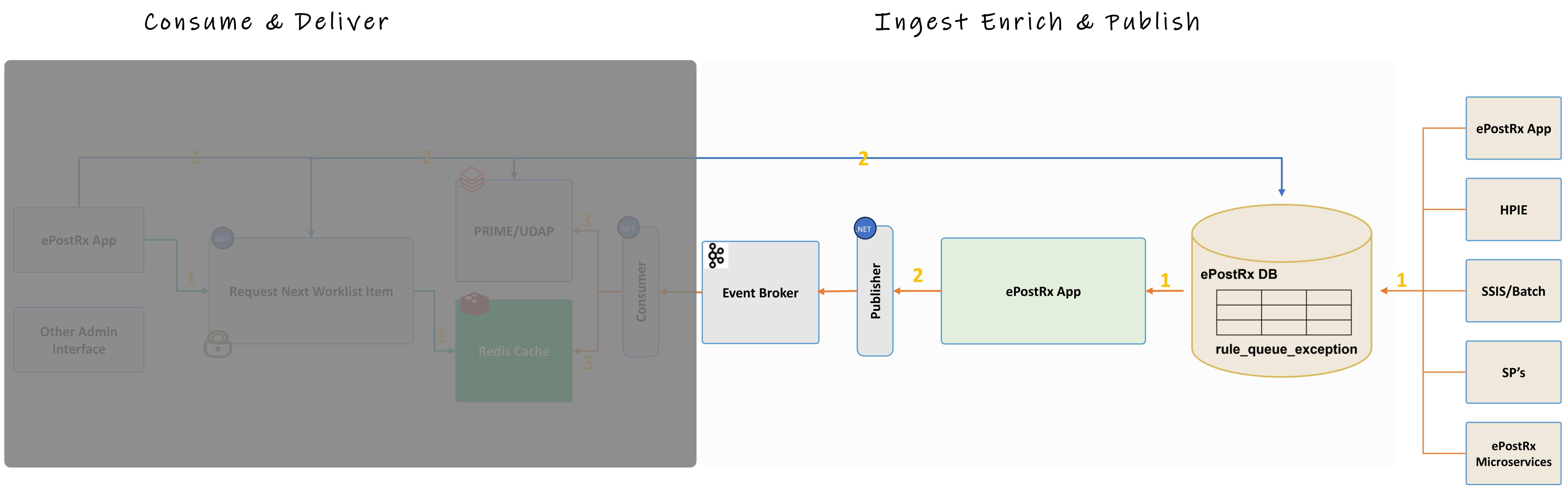
Next Button Architecture



Next Button Architecture



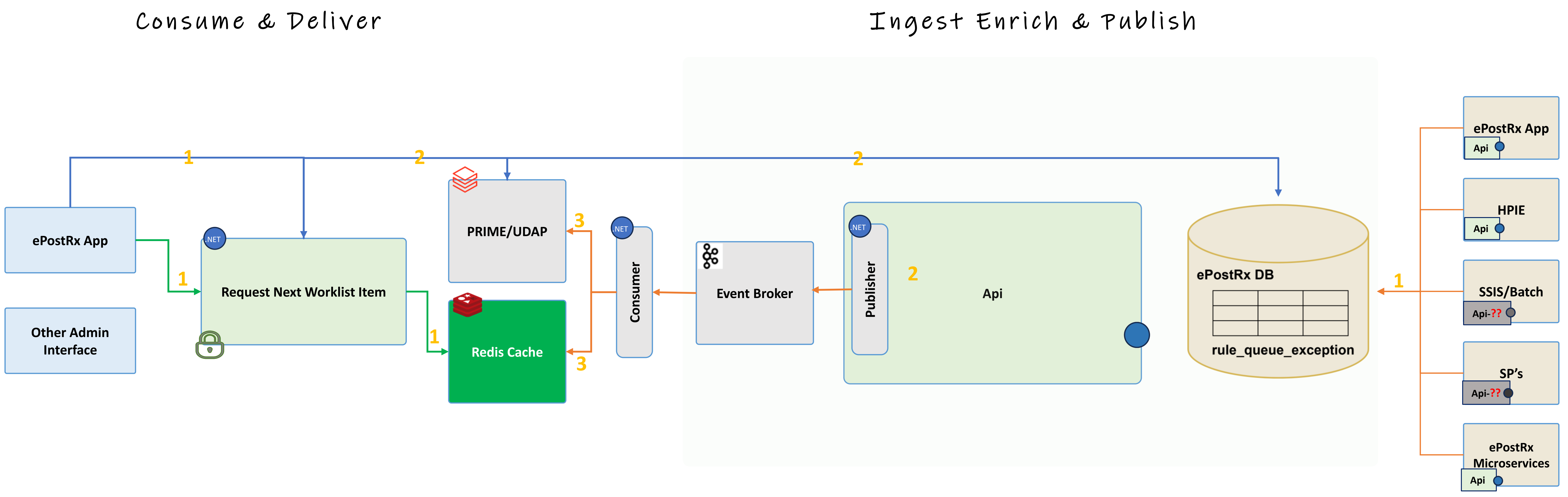
Next Button – CDC Solution Options

#	Option	Mechanism
1	**Microservice-based In-line Kafka Publisher**	HTTP call from apps/SPs → .NET service → Kafka
2	**Custom Polling (Business Tables)**	Poll business tables every 1s or less as needed
3	**Staging Table via SP + Custom Polling**	Apps/SPs call sp_staging → staging table Poll on staging table every 1s or less as needed
4	**Qlik Replicate**	Log-based CDC → Kafka
5	**Debezium + Kafka Connect**	OSS log CDC → topics + SMTs / Streams
6	**Staging DB (PostgreSQL) + Dual replication (Qlik)**	Qlik → Staging DB → Transform → Outbox → Kafka

Constraints

- **X - No MSSQL DB CDC**
- **X - No AZ Read Usage**
- **X - No Triggers**
- **X - No DB Intensive Agents**

Option 1 - Microservice-based In-line Kafka Publisher



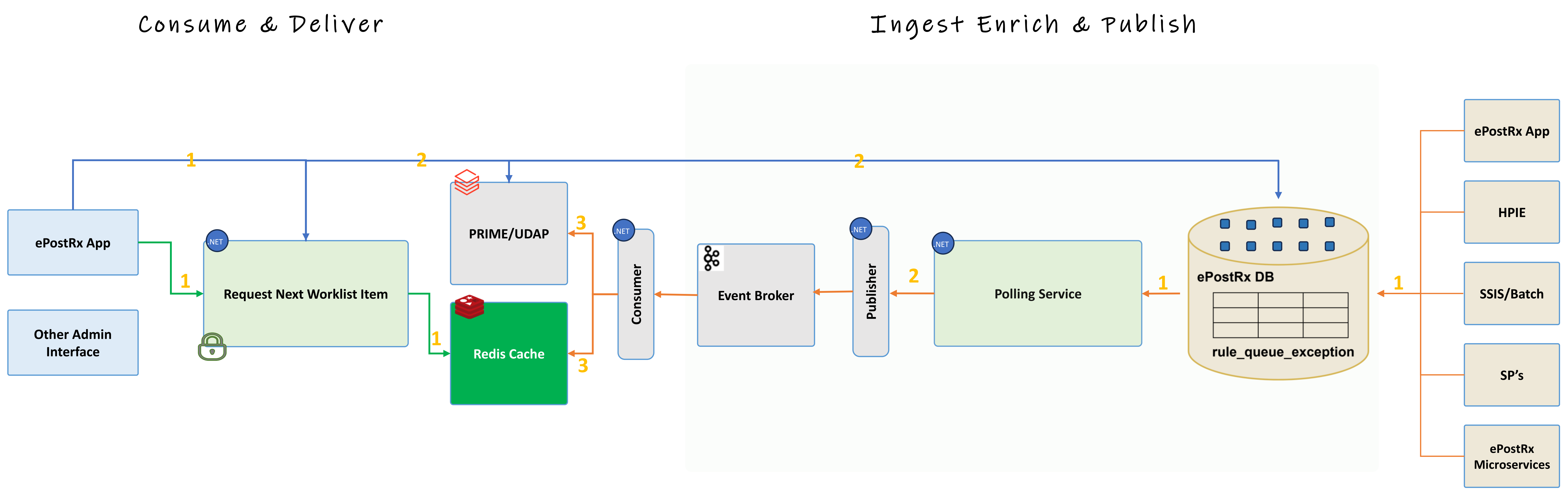
Pros

- ✓ Low -latency
- ✓ Real-time enrich and full control over enrichment logic

Cons

- ✓ Legacy components(Batch/SP's) can't easily invoke HTTP
- ✓ Transactional consistency limitations.
- ✓ Refresh/Reload Limitations
- ✓ Tight Coupling

Option 2 - Custom Polling (Business Tables)



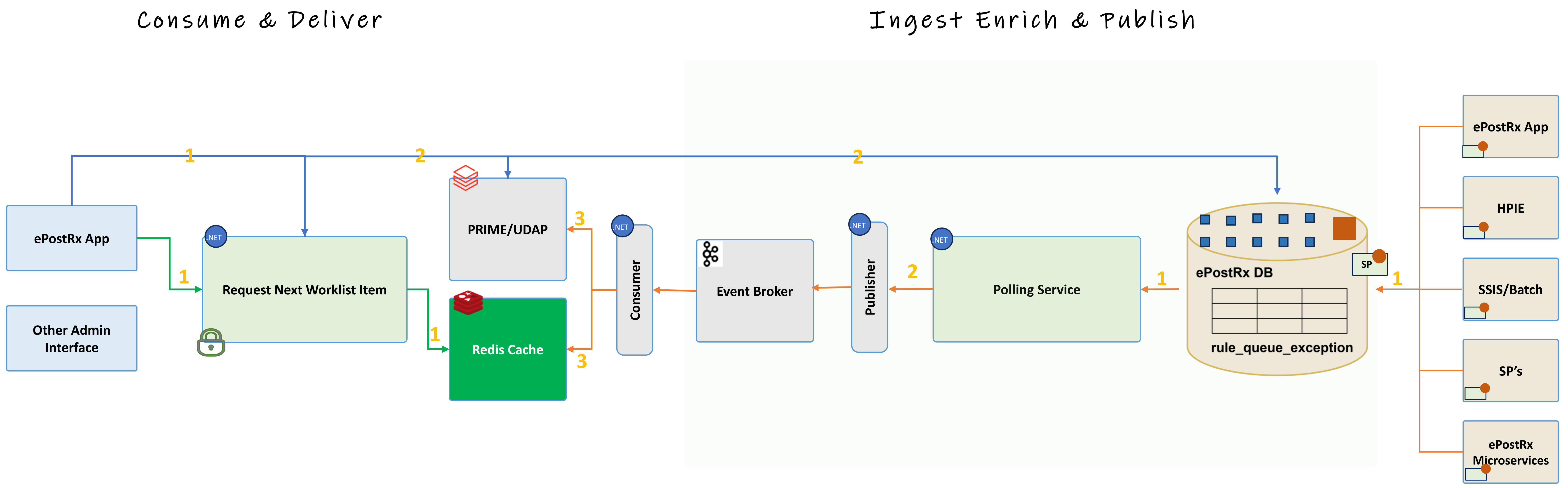
Pros

- ✓ Full control over the Polling logic
- ✓ No DB changes
- ✓ Centralized logic

Cons

- ✓ Very High DB impact
- ✓ High Dev complexity
- ✓ Quality Impacts
- ✓ Low latency requirements (<= 1 sec)
- ✓ Refresh/Retry limitations

Option 3 - Staging Table + Custom Polling



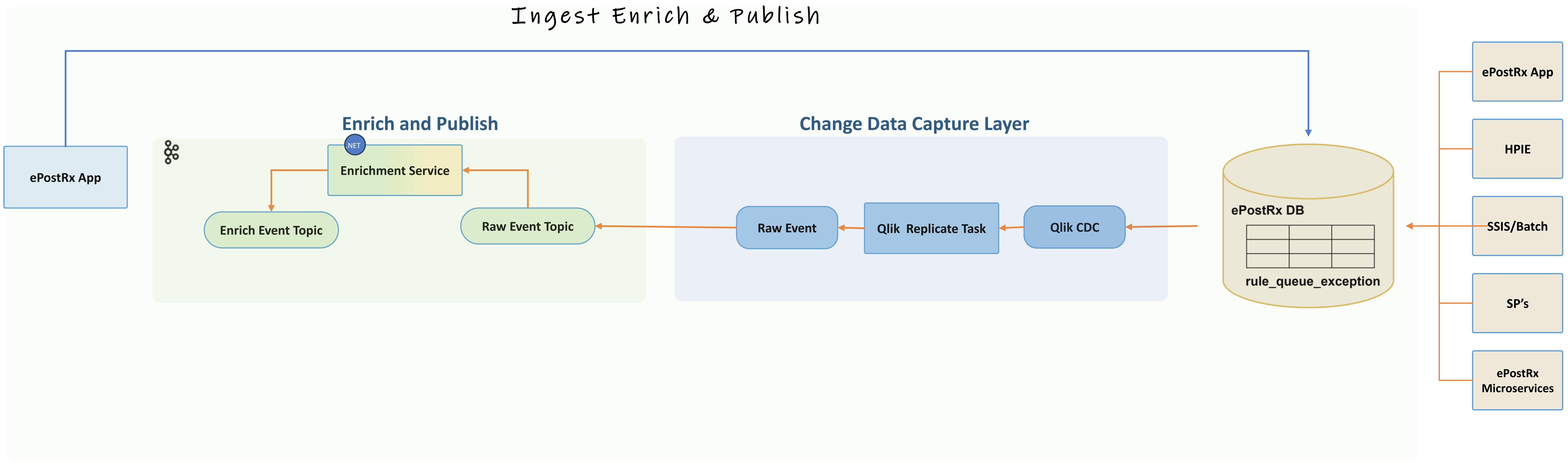
Pros

- ✓ Full control over the Polling logic.
- ✓ Relatively less complex polling
- ✓ No impact on business tables
- ✓ Centralized logic
- ✓ No Vendor/External Tool dependency
- ✓ Less Latency with required enrichment
- ✓ Clear separation of Concerns

Cons

- ✓ DB Growth
- ✓ Dev complexity
- ✓ Quality Impacts due to custom polling
- ✓ Additional maintenance and cleanup management
- ✓ Changes to the involved components.

Option 4 - Qlik Replicate - CDC



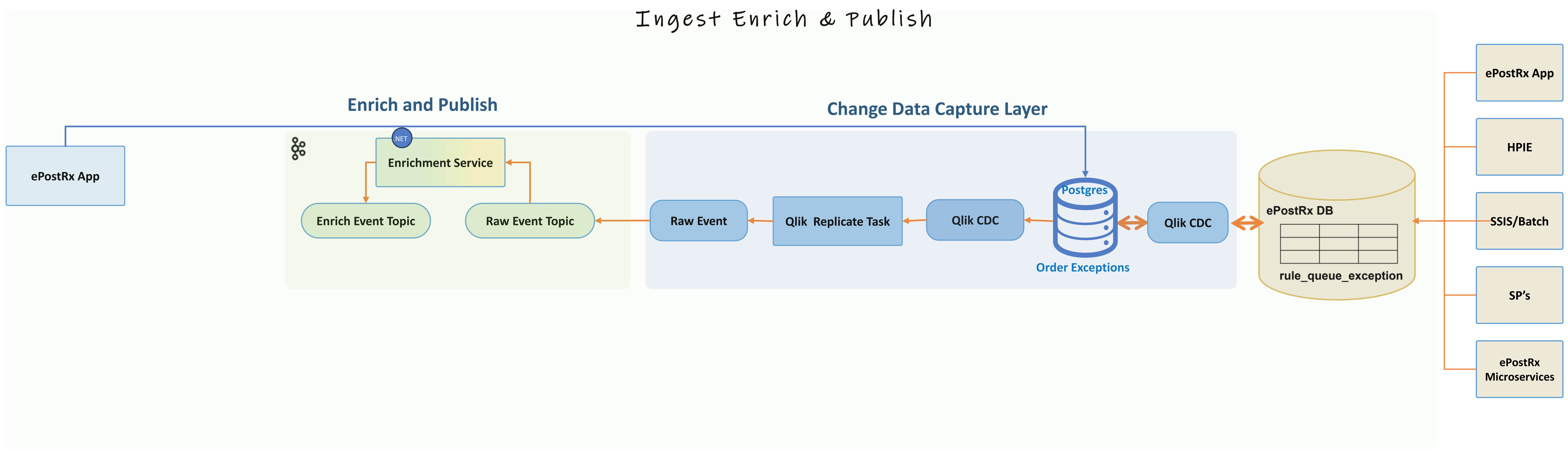
Pros

- ✓ Minimal impact on the source database.
- ✓ Matured Product.
- ✓ near real-time data flow.
- ✓ No Changes to external systems/components

Cons

- ✓ Licensing costs & Tool dependencies
- ✓ Limited enrichment capabilities
- ✓ Learning Curve
- ✓ Additional API calls for enrichment
- ✓ Per table CDC
- ✓ Additional Enrichment Complexity

Option 5 - Staging DB (PostgreSQL) + Dual replication (Qlik)



Pros

- ✓ No impact on the Master database.
- ✓ Matured Product.
- ✓ No Changes to external systems/components
- ✓ Easy Refresh/Reload
- ✓ Eliminates all DB dependencies and concerns
- ✓ Highly scalable
- ✓ Clear Separation of Concerns
- ✓ Enhanced flexibility and Extensibility
- ✓ Alignment with Cloud architecture.

Cons

- ✓ increased infrastructure
- ✓ extra hop
- ✓ Learning Curve
- ✓ License Cost
- ✓ Latency Impact
- ✓ Possible impact on Delivery timelines due to additional layers and technologies

Next Button – CDC Solution Options

#	Option	Mechanism	GO/No
1	Microservice-based In-line Kafka Publisher	HTTP call from apps/SPs → .NET service → Kafka	Not Recommended
2	Custom Polling (Business Tables)	Poll business tables every 1s or less as needed	Not Recommended
3	Staging Table via SP + Custom Polling	Apps/SPs call sp_staging → staging table Poll on staging table every 1s or less as needed	Recommended
4	Qlik Replicate	Log-based CDC → Kafka	Not Recommended
5	Debezium + Kafka Connect	OSS log CDC → topics + SMTs / Streams	Not Recommended
6	Staging DB (PostgreSQL) + Dual replication (Qlik)	Qlik → Staging DB → Transform → Outbox → Kafka	Recommended

Constraints

- X - No MSSQL DB CDC
- X - No AZ Read Usage
- X - No Triggers
- X - No DB Intensive Agents