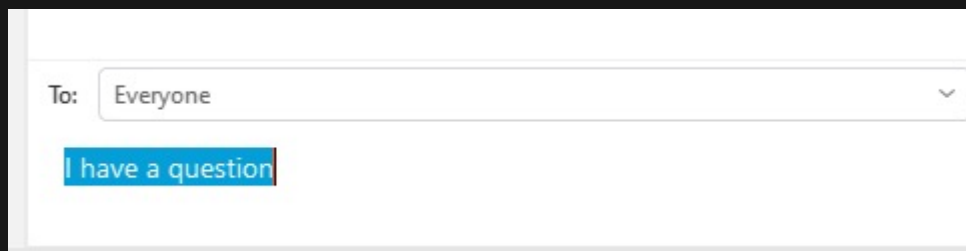
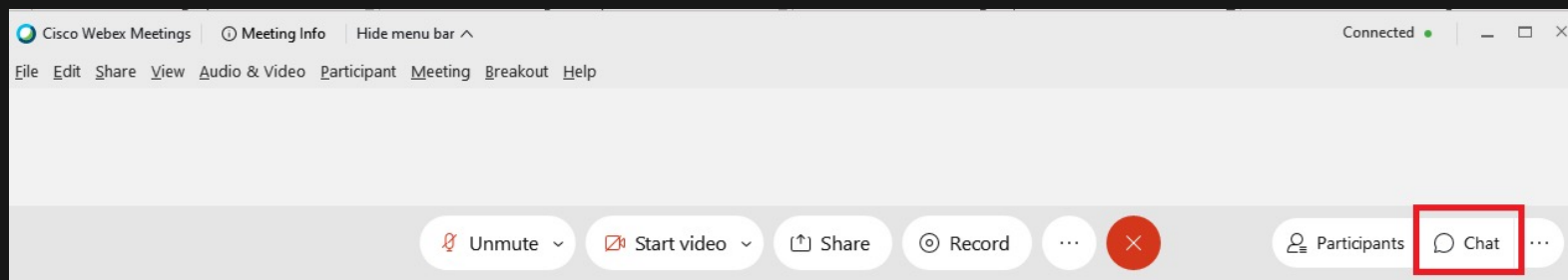


2021 Holiday Peak Readiness

IBM Order Management SaaS

- *Have a question?*



Speakers



Chris Burgess

Americas Team Lead –
IBM Sterling CSPO



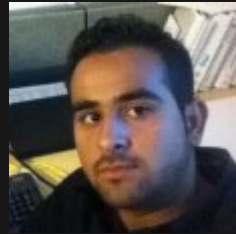
Mike Callaghan

Program Director –
Sterling Supply Chain Support



Aparna Subramanian

Technical Lead
/ Cloud Support Architect -
Order Management Support



Shoeb Bihari

Technical Lead / SRE Advisor -
Order Management Support



Senthil Ponnusamy

Monitoring Squad Leader -
Order Management Support

Journey to Peak Success

PLAN

- ✓ **Align Business and IT**
(projections, growth, NFRs, SLAs, channels, stores, catalog, inventory)
- ✓ **Product / platform**
(features, fixes, end user enablement)
- ✓ **Overlay Schedules**
(upgrades, releases, testing, freezes, peak sale events and times)
- ✓ **Best practices**
(implementation, config, testing)
- ✓ **Retrospective**
(prior peak events)
- ✓ **Identify Risks**

PREPARE

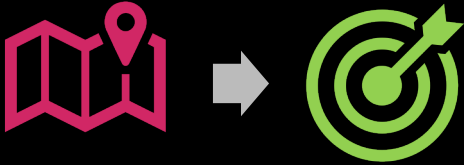
- ✓ **Performance test, tune**
(load, concurrency, data, integrations, store/call center user interactions)
- ✓ **Breaking points**
(infrastructure, app, middleware, integrations)
- ✓ **Failover, DR scenarios**
- ✓ **Ongoing housekeeping**
(DB optimization, purge, IBM fixes, clean logs, alerts)
- ✓ **Monitoring & Alerting ***
(continuous improvement)
- ✓ **Triage Techniques**
(isolate your bottleneck)

EXECUTE

- ✓ **People & process**
(roles & responsibilities, 24x7 schedules, escalation paths)
- ✓ **Job Prioritization**
(optimize schedules, disable non-critical processes)
- ✓ **Workload Prioritization**
(segregate key workload including BOPIS, backorders)
- ✓ **Runbooks**
(intervention, mitigation, throttling, emergency change)
- ✓ **Communication Plan**



OMS Holiday Readiness 2021



Our IBM Sterling Support mission is to **partner together proactively** to position you for success, quickly mitigate any issue that does arise.

We have worked relentlessly to establish a **stable platform** and robust collection of proven **best practices** for peak performance and stability.

We now have the opportunity to **shift our focus** to a deeper **commercial engagement** with clients needing additional help in proactive preparations.

How is IBM preparing?

- Ongoing product and platform improvements targeting performance, stability
- Validate resource capacity against production workload actuals
- Continuous improvement of monitoring, alerting, runbooks
- Operational and release freezes around peak season schedules
- Internal program to track, mitigate, communicate on at-risk accounts
- Mobilization of cross-functional SWAT team with for rapid engagement 24x7

How can IBM help you prepare?

- Leverage updated self-help best practices for guidance
- Perform platform readiness review of common config, shared components
- Prioritization of critical production issues or blockers
- **Enhanced Event Readiness offering** to deliver a comprehensive and prescriptive end-to-end peak readiness engagement with Expert Labs and Sterling Support Experts

Technical Best Practices

The IBM Sterling OMS Support team are continuously expanding our technical best practices based on the observations and learnings over our supported launches and peak events!



Are YOU ready? Review our prior best practices already available:

2020 Webcast Replays:

- Jul 9 - Peak Readiness Best Practices – [Replay](#)
- Sep 24 - Peak Day Mitigation – [Replay](#)
- Oct 20 – Best practice for a smooth Peak - [Replay](#)



Application & Database

- ✓ Configuration of APP, AGT, INT servers
- ✓ Housekeeping and maintenance
- ✓ Minimize contention, maximize concurrency
- ✓ Optimize long-running or expensive queries
- ✓ Transactional table size and purge validation



Integration

- ✓ MQ configuration tuning
- ✓ Batch feeds
- ✓ Call Center and Store operations
- ✓ Upstream eComm synchronous calls
- ✓ Inventory Visibility



Tools & Techniques

- ✓ Self-Serve Tool dashboards
- ✓ Representative load testing
- ✓ Find the bottleneck
- ✓ Understand your backlog
- ✓ Segregate, prioritize, throttle critical workloads

MQ Configuration

Recommendations below are to help mitigate risk against MQ message creation and consumption, which will only magnify under peak load

1. Review SST Performance dashboard and address any JMS-related exceptions before peak ([see KC link](#))
2. IBM will ensure **MaxChannels** is set to new default of 5000
3. Avoid using message Selector instead have dedicated internal/external queues.
4. Requests to IBM OMoC team for MQ config changes should be supported by results/validation from performance testing and tuning exercise
5. Avoid long running MQ transactions (*hitting default **MaxUncommittedMsg** limit likely indicates too many long running transactions*)
6. Review **MessageBufferPutTime** relative to **ExecuteMessageCreated** statistic from **YFS_STATISTICS_DETAIL** table for any slowness
7. Leverage Self-Serve Tool to find current MQ queue depth
8. Check for queues with very high queue depth (> 15000). Typically queues with large queue depth indicates the backlog and requires immediate action to clear the backlog
9. Ensure JMS properties **yfs.jms.sender.multithreaded**, **yfs.agent.bulk.sender.enabled** & **yfs.jms.sender.anonymous.reuse** are enabled.
10. Engage OMoC Support to generate alerts for specific queues at particular depth threshold

READ OUR BLOG! [MQ on OMoC - gearing up for the peak](#)

Key Lessons Learned

- **MQ Persistent Connection** usage and importance of **retry**
- **JMS Connections** and the potential impact on external system
- **MQ Failover** or network event does terminate in-flight and idle connection
- Understand behavior of applications connecting to IBM MQ when persistent connection are terminated
- Consideration around **reprocessable** service.
 - Reprocessable service can result in server staleness if there are erroneous messages with message length > 1 MB.
 - For erroneous message < 1 MB will be persisted in **YFS_REPROCESS_ERROR** table.
- For custom service, ensure JMS Sender properties are set correctly
 - Retry Interval (milliseconds) 100 ms
 - Number of Retries – at least 3 retries.

Are You READY?

- Are you aware of, and leveraging the new SST performance dashboard?
- What is your current MQ connection / Max Channels high watermark in use?
- How does this compare to **MaxChannels**? Do you have sufficient buffer?
- Are you receiving alerts for max queue depth for all relevant queues?
- Are you checking in on queues with very high queue depth (>15000)



Inventory Visibility

IBM Suggests you take on the following activities proactively:

1. Run a round of **supply sync** to ensure supply picture is updated at IV prior to the peak sales
2. Run **DG sync** a day or two prior to the peak sales, Use **syncDgAvailability = Y** as part of Update Distribution Group
3. Plan ahead to **avoid making changes to DGs** at peak time.
4. Set **safety stock** for items , this helps maintain a buffer inventory to ensure the channels do not over-promise
5. Ensure custom processes reaching out to IV are not generating **excessive tokens**
6. Have a **retry mechanism** for any 5XX errors being seen from the frontend systems. OOB adapter currently handles these errors using an auto retry mechanism set for agents and integration servers.
7. Avoid **payload limits** from being hit by ensuring the payload size that reaches IV from all channels is <500 KB
8. If using phase 1 IV adapter, ensure below properties are set-
 1. `yfs.purge.MergeDemandSupplyMultiRec=true`
 2. `yfs.inventorySnapshot.DemandSupplyMultiRec=true`

Maximizing IV usage

1. To enhance the performance:
 - Call IV directly to make reservations, as opposed to using `reserveAvailableInventory` from OMS
 - Reach out to IV directly for real-time availability (from front end)
2. Consider using Availability APIs:
 - For product details ,use the Get Node Availability Product Breakup API or Get Network Availability Product Breakup API (`/availability/{itemId}/network`) for the item in question.
 - For carts or orders, use the Get Node /network Availability API

Snapshots

1. Ensure that the front end has updated snapshot of availability by calling `ProductAvailabilityToSell.ShipNodeSnapshot` or `DgAvailabilityChange/publish_all` few hours prior to peak sales
2. Use the new Jobs API to understand which jobs are in progress (like DG sync, `ProductAvailabilityToSell.ShipNodeSnapshot`) before triggering another job
3. If using COS, request access to the cloud-storage-tools utility which will help with download/deletion of events from COS buckets

Real Time Availability Monitor (RTAM)

1. **Full sync mode is expensive**, should be scheduled only during low-traffic hours (ie. during night). For peak season , careful planning needs to be done around business events and planned traffic
2. **Preferred methods** to push inventory availability out to external system are **Activity-based** and **Quick-sync** mode
3. **For Activity-based RTAM**, validate activities created by node capacity changes
4. **Optimize RTAM performance** with this high level configuration:
 - `yfs.jms.sender.multithreaded`
Enable when publishing alerts to external queue; allows RTAM agent threads to publish message parallelly
 - `yfs.yfs.rtam.readInventoryForOnlyActivityNodes`
Implicitly enabled based on conditions:
 - a. Running activity based RTAM
 - b. Monitoring item availability at node level and network level.
 - c. Property not explicitly disabled (i.e. `yfs.rtam.readInventoryForOnlyActivityNodes=N`)
 - d. Monitored items is not a bundle.
 - e. All DGs at network level are monitored at node level
 - **Number of Records To Buffer** can be increased to 25K in reference to `MessageBufferPutTime`
 - Disable `Compute availability information with Ship Dates for RTAM` flag (`COMPUTE_AVAILABILITY_FOR_RTAM = N`) if availability information is not required.
 - Optimize Event template to remove unnecessary attributes

Hot SKU / OLA

Legacy

1. Hot SKU & OLA feature are enabled by default in OMoC.
2. Review and tune Hot SKU factory values based on expected order profile.
[KC Link](#), [Community article](#)
3. Review `INV_INVENTORY_ITEM_LOCK` table during performance tests and pre/post sales to assess the level of contention and items involved.
4. With OLA enabled, there will be granular level of locking on `YFS_INVENTORY_DEMAND` record (item, ship node, demand level)
 - During peak time, avoid running inventory synchronization which updates `YFS_INVENTORY_DEMAND` along side `YFS_INVENTORY_SUPPLY`
 - Configure RTAM monitoring to generate alert to replenish the inventory or mark the item unavailable

Capacity

- Enhance the promising APIs performance by enabling capacity availability agent. Additional Guidance: [KC Link](#), [Blog coming soon!!](#)
 - This is a time-triggered agent that calculates the capacity upfront and stores in the `YFS_CAPACITY_AVAILABILITY` table from where the data can be read during promising calls.
- Depending on the business objective consider following property `yfs.nodecapacity.lock=N` to avoid locking during capacity check.

Smart Sourcing

- Reading and processing inventory for nodes that do not contain inventory is costly.
- To improve performance, smart sourcing can be used to dynamically determine the nodes to consider for sourcing product items. [Smart Sourcing](#)

Agent/Server Configuration

The following configurations have repeatedly been recommended to clients to help address known performance, stability, scalability issues:

- 1. Tune memory parameters** based on analysis from GC logs ([KC link](#))
 - For NextGen platform, ensure appropriate profile is selected based on testing: Balanced, Compute, Memory
- 2. Optimize schedule** of batch processes around expected peak loads
 - Ex. Complete Inventory Sync before the peak hours begin; Add more JVMs in the wee hours of the peak day and complete the process well before the rush
- 3. Disable non-critical agents** to reduce unnecessary contention and usage of resources by disabling any processes that are not business-critical during peak
 - Disable IBA which is known to be intrusive, and best to avoid peak usage
 - Explore option of disabling non-critical Order/Shipment Monitor rules
 - Configure and run Close Order/Shipment agent before peak season ([link](#))
- Avoid having multiple agent criteria in same agent configuration; makes for easier troubleshooting.
- Ensure naming convention of agent/integration server accurately represent function which it performs; makes for easier troubleshooting and assessing the impact
- Reduce message payload by optimizing API, event templates, pull only required data (set `TotalNumberOfRecords` to restrict output)

Performance Profiles

NextGen

There are three server performance profiles for the agent or integration server.

- 1. Balanced:** Provides moderate memory and computing power for typical OMS workload.
- 2. Compute:** Provides additional computing power and moderate memory. This type of profile is more suitable for workload like RTAM, etc.
- 3. Memory:** Provides additional memory and moderate computing power. This profile type is more suitable for purge agent.

Which profile should we use?

There is no definite formula to identify the right profile. However, you can use the following guidelines to select a performance profile for optimal results.

- Start with a single thread for the server, single instance of the server, and the **Balanced** profile.
- Increase the number of threads gradually to arrive at the correct profile and maximum number of threads per server.
- If CPU or memory allocation does not change significantly with each additional thread, continue with the **Balanced** performance profile. Servers that spend most of the time calling external services display this kind of resource use pattern.
- If the JVM heap utilization stays around 80% or increases significantly with each additional thread, change the profile to **Memory**.
- With any of the performance profiles, if the CPU allocation stays around 70% or memory allocation stays around 80%, you might scale the server (Pod) rather than increase the number of threads.

Guidelines [for selecting the performance profile to improve server performance](#)

- Self Service Monitoring Dashboards – JVM, DB, JMS, API and Service performance.**
- OMS SMC Console – Monitoring cache statistics and making thread changes.**
- Query YFS_STATISTICS_DETAIL table for API and Service performance stats.**



OBJECTIVE: Selecting an optimal performance profile

Next generation



PROBLEM: Target to achieve 30k TPH for createOrder w/ 2.5 average lines

- Select optimal server profile and thread configuration for createOrder integration service to ensure service can scale w/ custom logic and configuration.



APPROACH:

Configure create order integration server in OMS

- Initial Threads =1
- Performance Profile = Balanced
- Default # of JVM Instances = 1

Execute and monitor the server performance via **Self Service Tool**

- API Response time (ms)
- Invocations (rpm)
- Container CPU Utilizations
- GC CPU Utilizations
- JVM Heap Utilization
- Order Lines Throughput

Observe and Adjust the configuration until throughput is achieved

- Increase the # of threads
- Switch Performance Profile
- Increase the # of JVM instance

NOTE: Below numbers represent OOB createOrder with some customization

Server Type	Integration	Performance Profile	Balanced				
Server Name	CreateOrderServer						
JVM Instances	No. of Threads Per JVM Instance	API Response (ms)	Invocations (rpm)	Container CPU %	GC CPU %	Heap Utilization	Order/Hour (2.5 average lines)
1	1	198	305	51.5	3	52.7	18760
1	2	285	420	85	7	57.8	25200
1	3	410	432	96	12	62.4	25920
2	4	580	804	99	19	70	47398

Server Type	Integration	Performance Profile	Compute				
Server Name	CreateOrderServer						
JVM Instances	No. of Threads Per JVM Instance	API Response (ms)	Invocations (rpm)	Container CPU %	GC CPU %	Heap Utilization	Order/Hour (2.5 average lines)
1	1	182	324	25.6	1.4	29.1	19440
1	2	196	612	47	3	35.5	36720
1	3	219	810	61	5.87	44.2	48600
1	4	230	1041	75	6.47	51.7	62100

Optimal Solution:

- ✓ Threads = 3
- ✓ Performance Profile = Compute
- ✓ JVM Instances = 1



RECOMMENDATIONS:

- Spawning additional (untuned) instances of agent to try and improve throughput let to **exhaustion of resource** allocation available
- Review KC Guidelines to select performance profile** | **Review community article on Sterling OMS Performance Profiles**



OBJECTIVE: Avoid Message Selectors



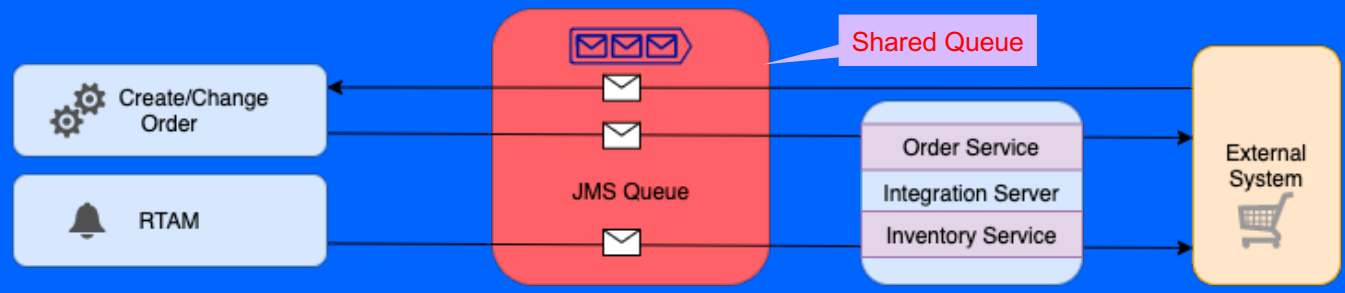
PROBLEM: OMS is experiencing downtime in Production !

- Significantly high backlog for some of the order process integration queues, createOrder not moving / extremely slow for several hours
- RTAM server was stuck for several hours resulting in impacting inventory picture on eComm and other channels



OBSERVATIONS:

- Consistently high CPU observed on MQ server.
- Identified that primary contributor of MQ high CPU / slowness was due to use of MQ 'selector' to pull specific message types from a shared queue
- In one case, shared queue used for WCS-OMS integration reached 800K messages, both WCS/OMS were PUT/GET to same queue as part of process



- Issue was resolved by eliminating the use of message selector by separating the queues within the integration pipeline.



RECOMMENDATIONS:

- Do not use message selector where equal distribution of workload is not guaranteed
- Understand transaction flows, timings and high watermarks



OBJECTIVE: Avoid Server Staleness

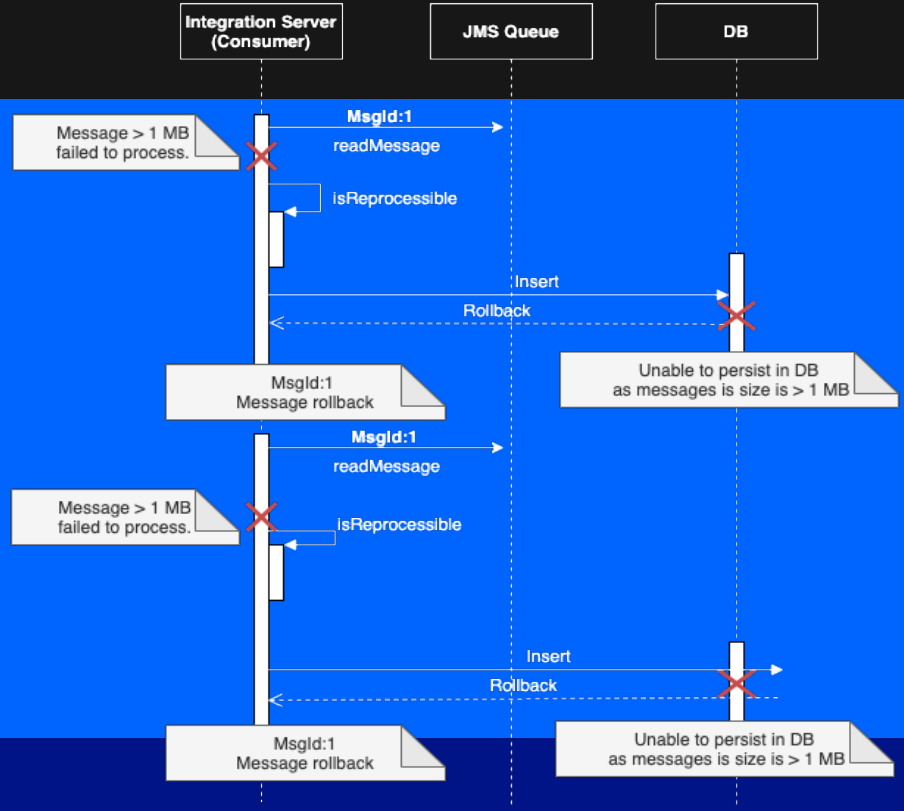


PROBLEM: Orders not being processed in Production !



OBSERVATIONS:

- Excessive Errors noticed as part of proactive monitoring from `createOrder` Integration Server:
 - `ErrorCode="YFC0001" ErrorDescription="Record already exists in the database."`
 - `ErrorCode="YDB92_001" ErrorDescription="The data bound to the column is invalid in this context."`
- Reprocessable errors flag set to Y
- Message size >1 MB in corresponding queue, Was unable to insert on `yfs_reprocessable_error` table as MESSAGE size exceeded 1 MB
- Server running in single threaded mode
- Mitigation – Removed “bad messages” manually from queue



RECOMMENDATIONS:

- Exercise caution while accepting payloads for integration servers with `reprocessable` errors flag set to Y
- Look into increasing threads on integration server as interim relief

Performance Testing Guidance



Performance testing is an art, but a mandatory one! It is imperative to vet out issues in advance on pre-production load testing, rather than wait for it to surface as a business critical production issue!

Refer to Knowledge Center for detailed Tuning and performance guidance.

- 1. Projected peak volumes** – Ensure business and IT are in sync on expected peak loads to ensure planned tests are accurate. Test for forecast of peak orders, orderlines, common cart skus/items, and real time inventory calls
- 2. Representative Combination Tests** – Assemble components to reflect real time DATA, scenarios and run in parallel to ensure adherence with NFR; Stage data for various components and run them under full load (ie. Create + Schedule + Release+ Create Shipment + Confirm Shipment + Inventory Snapshot (IV))
- 3. Agent and integration servers** – ensure asynchronous batch processing components are tested in isolation and in combination with broader workload; ensure to tune agents (processes, threads, heap size) to meet expected peak SLAs/NFRs on throughput
- 4. Test Failure Scenarios** – validate resiliency of overall system and operations, ensuring graceful recovery if front-end channel (web, mobile, Call Center, Store, EDI, JMS), backend OMS, or external integration endpoints fail. Include ‘kill switches’ in any components that can be disabled to avoid magnifying an isolated issue into system wide one, especially for any synchronous calls.
- 5. Confirm Peak days and Hours** - Share any specific key dates or max burst times with IBM Support, including code freezes, flash sales.
- 6. Coordinate with IBM** - Inform IBM (CSM/Support) in advance when load tests are planned if any data or diagnostics (such as against Database) are needing to be captured; IBM can also then review internal metrics and response in parallel.
→ **Inform IBM in advance of major configuration changes (sourcing rule: Increase in SFS due to COVID).**

Ensure you are clear on the actuals vs. projections of to facilitate accurate tests, and for IBM OMoC team to validate sufficient resources

Metric	2020 Peak	2021 YTD Peak	2021 Proj. Peak	2021 Load Test Peak
Orders / hour (max)	?	?	?	?
Orderlines / hour (max)	?	?	?	?
Get Inventory Availability	?	?	?	?
Reserve Inventory	?	?	?	?
Inventory Adjustment trickle	?	?	?	?
Inventory Adjustment burst	?	?	?	?
Concurrent Store/CC users	?	?	?	?

- Are you leaving sufficient buffer between load tests and holiday freeze?
- Are performance tests representative of volumes (order lines, real time sync calls), plus:
 - Asynchronous batch processes (agent and integration servers)
 - Store (Pick, Pack, Ship) and Call Center concurrent user activity. **Validate all navigation flows.**
- Do your performance test runs cover any potential common cart item, to determine if further configuration changes may be needed to avoid contention:
 - Realistic inventory availability picture
 - Orderlines to include expected commonality of free gift / hot items
- Performance validation of initial data load related to peak. (Migrating Orders, Catalog, etc)
- Initial Inventory synchronization to IV.
- Will the endpoints external to OMoC be able to scale (rate limits, performance)?
- Adoption of Self Serve Tools & subscription to alert and notification.

Position for Peak Success



To best position for success on the OMS platform, it is important to understand how your application handles various scenarios known to challenge performance or stability. Testing in pre-production with data/workloads representative of production enables ability identify and address issues without impact to production business and operations.

1. Resource/Hardware **sizing** based on segment profile, but is validated as OUTCOME of performance testing, not a replacement for it
2. **Database** is common bottleneck, not due to capacity, but untuned queries, missing indexes, competing processes, unqualified end-user searches
3. Underlying config **data** has significant impact on performance, including database query execution plans, inventory sourcing rule evaluation
4. Accumulation of **transactional data** over long periods of time (and failure to purge as possible), may degraded query performance
5. **Item distribution** and commonality must reflect realistic peak load; high-demand / hot items (free-gift) may significantly impact concurrent processes
6. Composition of a custom service (**Service definition framework**) can lead to inefficient execution or potential lock contention, reducing throughput
7. Understanding **queueing**/de-queueing rates to align with business SLA / expectation (ie. create order, confirm shipment); SI needs to know when there is an issue to intervene / troubleshoot (ie. particular queue depth)
8. Agent/integration server throughput must be sufficient, but remain below **max resource allocation**; varies on number of instances, server profile
9. Important to understand / validate impact to upstream application (eComm) if specific synchronous calls into OMS slow or become unavailable



Real Scenarios (*Real impact...*)

- COVID-19 led to more stores enabled for BOPIS which made DG significantly larger, which led to more time for synchronous inventory availability calls; similar scenarios where client had to split nodes in DG to improve throughput
- Rapid ramp up of **in-store associates** led to several unqualified searches in Store and Call Center apps which caused significant overall degradation
- multiAPI made 8 successive API calls led to poor response, needed to be refactored to **use asynchronous** requests (via MQ to drop message on queue)
- **Custom service call** to getOrderList API was missing OHK in input, each invocation caused fetch of 5K records which led to a crash, had to limit records
- Needed to **throttle down** instances/threads of agent to reduce concurrency contention issues (Create/Schedule/Release) and optimize throughput
- Gradual **memory leak** led to out-of-memory condition after a couple days; similarly untuned heap led to excessive GC overhead, high CPU, slowness
- Daily manual processing of orders via java client against single JVM bypassed load balancer and overwhelmed JVM to OOM/crash
- Upstream eComm site was unable to gracefully handle a short period of unavailability from backend OMS and took hours to recover
- Unintentionally carrying capacity for high volume node during the peak. (Example: Pop up /Temporary fulfilment warehouse)
- Avoid changes to DG in IV during peak time

Understanding the problem – well defined problem is easy to resolve (mitigate)

Timelines, Impacted components, Diagnostics captured, Business impact play an important role in the road to system recovery.

Application / Agents/ Integration Server Health

- Issue and alerting condition
- Measure of degradation and business impact seen – Performance or Functional?
 - **Performance:** Are all APIs slow or particular (e.g. Inventory Lookup / Reservation vs. Order status updates / order lookup)? Is it intermittent or always slow?
 - **Functional:** Is this issue reproduceable? Can issue be observed outside the monitoring app? Is there an error? few errors or excessive? How frequent?

Timing/Timeline

- When did the issue start?
- Any patterns observed with other processes or under specific load?
- Does issue worsen over time?

Impacted Components

- Are only certain servers affected?
- What are all dependent services/component for the servers? (External system, MQ, SMTP)

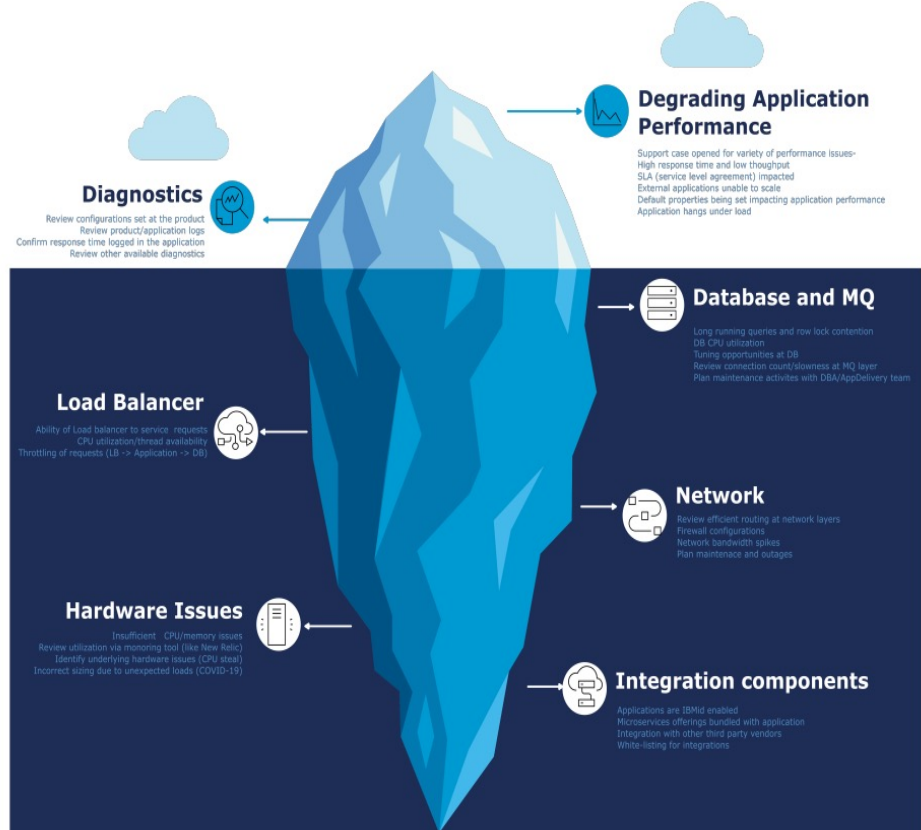
Be prepared for peak days!

- Confirm focal, maintain internal contact sheet.
- Define a support schedule (24x7 peak-hour coverage)
- Setup a communication plan.
- Document escalation criteria and procedures
- Ensure activities by business, performance, operations are in sync

Contact IBM Support in a crisis

- Contact IBM Support to open a severity 1 Case
- Escalate case by new Escalate button in the Support Portal

ICEBERG EFFECT Supporting IBM Sterling Order Management on Cloud



Proactive Support Model

- IBM performs the following types of monitoring for assessing the health of your production site and its services:
 - **System and infrastructure** - checks health and well-being of physical server hardware, virtual machine resources, and network [KC link](#)
 - **Application** - application JVM, application server node instances, database logical servers, and application components. [KC link](#)
 - **Synthetic** - customer's application can be opened from the browser and whether the customer's server can be pinged. [KC link](#)
- Continuous improvement of alerting based on lessons learned across infrastructure, database, app servers, agent/integration servers, error conditions
- Continuous improvement of internal runbooks to quickly mitigate issues

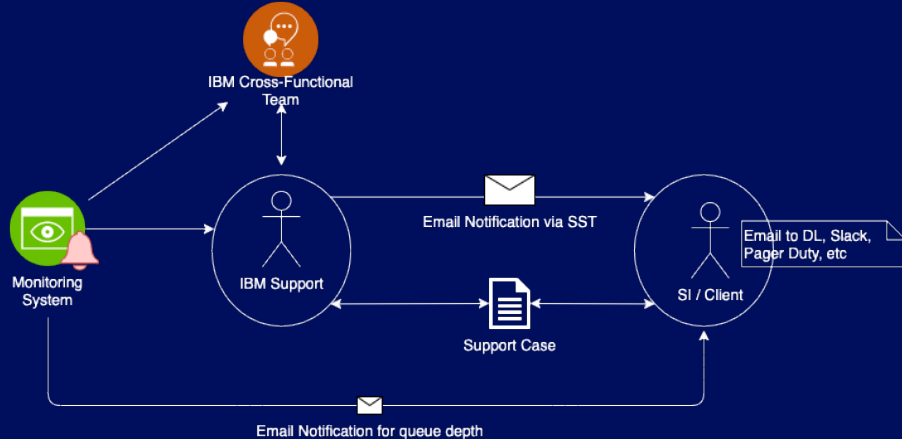


IMPORTANT!

Self-manage notification distribution lists via SST to receive product notifications.

Update your IBM Support Portal profile with your phone number in case we need to engage you!

Proactive Notifications



EMAIL (✉) for issue with Multi-tenant shared infrastructure component

- If there are issues from a shared component within data center such as firewall or network, an **email notification** would be sent to all potentially impacted clients as soon as IBM has determined the potential scope / impact
- Open a case for specific inquiries during incident for additional detail on root cause

SUPPORT TICKET (🎫) for issue with client-specific component

- If there are issues specific to a client, whether infrastructure or application, IBM Support will proactively open a Severity 1 **Support case** within the IBM Support Portal, informing the client of the situation, and in some cases either ask for approval to take a specific action, or request that the client take an action or review



Production Alert Handling

IBM are continuously improving monitoring, alerting and runbooks to allow quick handling of production issues:

1. **Proactive case** will be opened by IBM Support to inform client/SI of triggered alert, and that investigation is underway
2. **IBM capture diagnostics**, review, determine source of alert
3. **IBM take action to mitigate** if possible, inform / get consent from client/SI as needed (such as restarting an agent/integration server)
4. In event client/partner need to **take action**, the proactive case will be used to convey this information

Application

MQ Connectivity issue

(Max connections, JMS Transaction Failures)

- ✓ Critical MQ connection issue → 5 JMS exceptions in 5 min.
- ✓ Excessive MQ Connection Reset - MQRC_CONNECTION_BROKEN → 10 in 10 min.
- ✓ MQ - Invalid Message → 2 in 10 min.

Database Connectivity Issue

(Max connections, DB Transaction Failures)

- ✓ Excessive Database Query Timeouts (YFC0006) -> 100+ timeouts in 30 min.
- ✓ Critical DB connection issue (YFC0003 - Database Error) -> 5 exceptions in 5 min.

Application Server

(Real Time Failure / Sync Calls)

- ✓ GC (Global) Overhead (High) → 5% for 10 min.
- ✓ Heap memory usage (High) → 80% for 15 min.
- ✓ Server Hung/Unresponsive → 90% threads used for 5 min.
- ✓ Excessive Errors by JVM - critical → 20% error rate for 10 min.
- ✓ Server Startup Failure (YIC10004) - Cache Initialization → 1 or more occurrences
- ✓ Excessive REST:HTTP 401 errors → 30 in 5 min.
- ✓ Process DOWN (Health Check , Missing POD)

Stale Agent/Integration Server

(Stale Agents)

- ✓ Heap memory usage (High) → 80% for 60 min.
- ✓ GC (Global) Overhead (High) → 5% for 10 min.
- ✓ **Custom Queue Depth Alert**
- ✓ Agent & Integration Process DOWN

Stale/Stuck Query

- ✓ DB: Lock-Wait
- ✓ DB: Long Running Query
- ✓ DB: Not enough storage is available, SQLCODE=-973 → 1 or more occurrences
- ✓ DB: Too many open statements, SQLCODE=-805 → 1 or more occurrences

Network/Connectivity Failures

- ✓ Connectivity Issue (ConnectException, SocketException) → 3 in 10 min.
- ✓ Data Extract Failure - YFS: SFTP server is not reachable

OOB Integration

- ✓ IV Integration Failures (Connectivity issue & Error Response)
- ✓ SIM Integration Failures (Connectivity issue & Error Response)

MQ Server (Active/Passive)

- ✓ MQ Listener Down: No listener running for (OM_QMGR)
- ✓ MQ Server Down
- ✓ Generic Queue depth alert 100K/30%

Low Severity alerts

(Based on Exceptions)

- ✓ Data Extract Failures, monitored for ErrorCode: CDE100005, CDE100014, CDE100016, CDE100019, CDE100020
- ✓ JMS: Queue not created Error: `javax.naming.NameNotFoundException`
- ✓ JMS: Queue connection configuration Error: `javax.naming.NoInitialContextException`
- ✓ DB: Inserted Column Data > Column Size, Error: YDB92_001 (10+)
- ✓ DB: Failed Update due to concurrent modification, Error: YFC0009 (100+)

Infrastructure

Database Server

- ✓ Database CPU, Disk Utilization
- ✓ Host is not responding for 5 minutes.
- ✓ Transaction logs size
- ✓ HADR/TSA connection
- ✓ DB Read/Write/Disk Utilization

Other VM Host

- ✓ Local , NFS Disk Utilization
- ✓ VM (host) is not responding
- ✓ CPU, Memory , Disk Usage
- ✓ CPU Steal
- ✓ Cluster Health

Synthetic

Order Management Components

- ✓ Availability Check

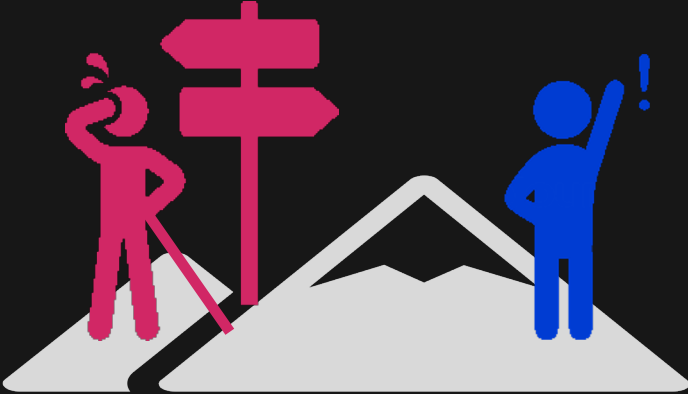
Inventory Viability APIs

Self-Serve Tooling

Business User Control

Store Inventory Management APIs

How can IBM Help you prepare?



IBM Internal Readiness Review



IBM OMoC team is taking **proactive steps** of platform and infrastructure to ensure readiness of our IBM SaaS solutions



Production Operations

- ✓ Monitoring, alerting policies, internal runbooks
- ✓ Resource utilization review across application and DB
- ✓ Proactive Support process and notifications
- ✓ Raise proactives (long running queries, high error rate)
- ✓ Maintenance and release freeze



Platform

- ✓ Resource capacity review based on production actuals
- ✓ Standard MQ config validation
- ✓ Database maintenance scheduling & optimization
- ✓ SaaS component integration checks (IV, OrderHub, etc)

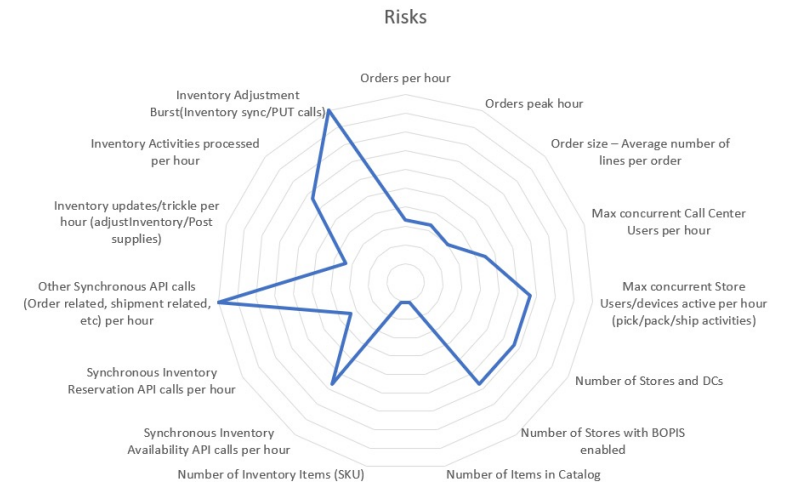
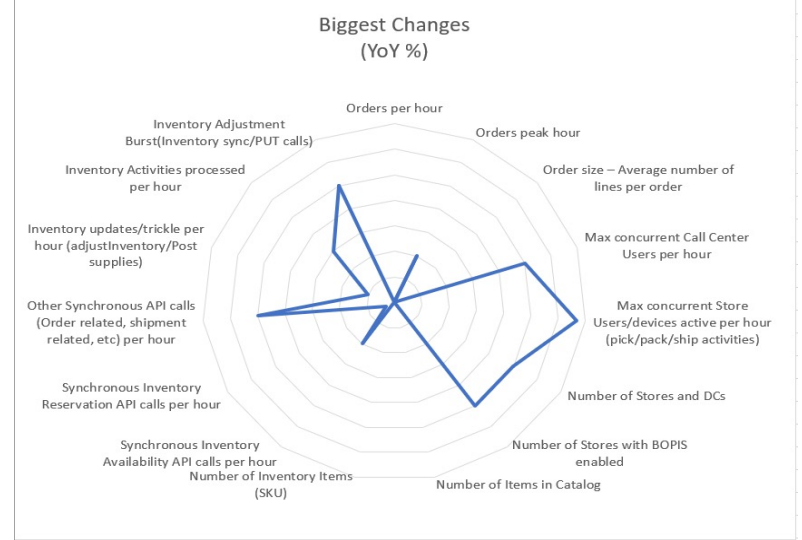
2021 Risk Self-Assessment



IBM OMoC team recently released our annual Holiday Readiness Questionnaire.

Take the opportunity for self-assessment and jumpstart the conversation on how IBM can help with your concerns!

Holiday Readiness Risk Self-Assessment ibm.biz/OMS-EventReadiness-Qs



IBM AI Applications – Enhanced Event Readiness

- Assigned team of technical experts engaged before, during and after your event (up to 3 months)
- Support focal to walk you through proven preparation and planning exercises
- Built-in IBM Expert Labs health check and performance assessment covering Application, Database and Infrastructure layers
- Elevated oversight and management of critical customer support cases
- Enhanced communication via scheduled touchpoints and status reports
- Retrospective and lessons learned session post-event.



IBM® AI Applications Enhanced Event Readiness helps businesses strengthen their technology infrastructure ahead of known events to help prevent unexpected disruptions and deliver a seamless customer experience.

Make your biggest days of the year your best days of the year!

Enhanced Event Readiness - Offering

IBM Sterling experts are available through a 3-month comprehensive technical engagement to help proactively prepare for peak season success on the IBM Order Management solution.



Contact your IBM CSM or account team for details!



Performance Health Check

IBM Expert Labs Performance Expert will engage performance health check and provide a comprehensive technical review of application, database, infrastructure and provide a set of recommendations to best position for peak performance and stability.



Proactive Support Consultation

A proactive approach to peak success with prescriptive recommendations based on our vast experience and consolidated best practices in performance, stability, and peak volumes.

Senior Technical Support experts from the Performance Squad will also be available for consultation throughout event preparation and execution.



Support Advocacy

Named IBM Support Advocate to help oversee the offering engagement, reporting, tracking and completion of recommendations, and client readiness discovery sessions.

Ongoing coordinated communication across engaged IBM parties and provide elevated oversight and prioritization of critical support cases and management visibility regarding your event/planning

Enhanced Event Readiness - Activities

IBM Sterling experts are available through a 3-month comprehensive technical engagement to help proactively prepare for peak season success on the IBM Order Management solution.



Contact your IBM CSM or account team for details!



Performance Health Check

- ❑ **Application tier review** including analysis of application, agent, integration server configs, error logs, potential transaction bottlenecks
- ❑ Identify potential **risky integration points** for synchronous and asynchronous calls,
- ❑ Review maintenance, monitoring, housekeeping
- ❑ **Database tier review** including evaluation of health metrics on both transactional and non-transactional tables, DB monitoring (locking), DB configuration
- ❑ **Overall infrastructure health check** including analysis of resource usage reports, capacity planning, in-depth technical stack review (as needed)



Proactive Support Consultation

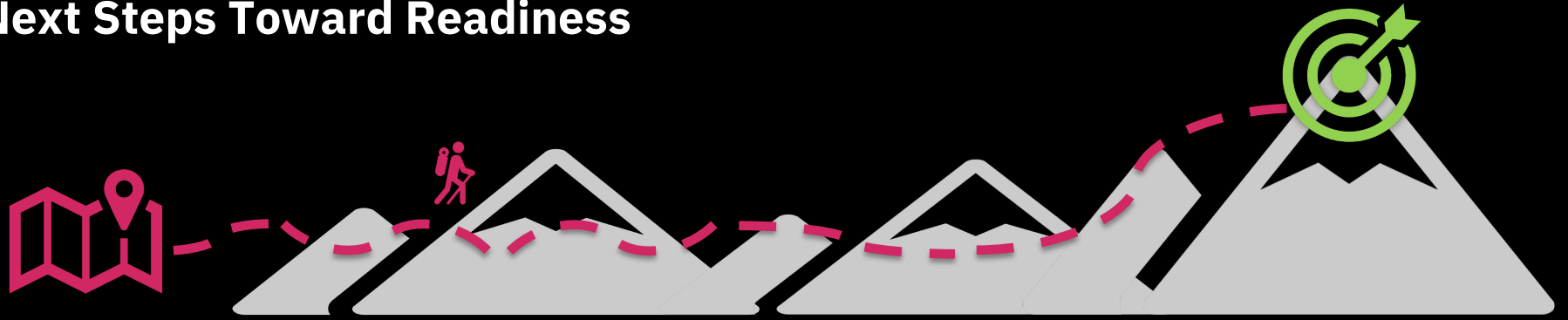
- ❑ **Technical enablement** including best practice configurations, proactive housekeeping and maximizing value from SST tooling
- ❑ **Proactive consultation** including testing coverage, techniques to prioritize workload, preparation of technical runbooks
- ❑ **Validation and recommendation** of application configuration to align to proven best practices
- ❑ **Review** potential risk based on production operation, monitoring, utilization, projected workloads
- ❑ **Retrospective** to ensure closed loop on historical critical Support cases, prod alerts
- ❑ **Stand-by Support** for critical business periods to engage on beyond the typical scope of Standard Support to critical triage efforts



Support Advocacy

- ❑ **Planning** including assist your Solution Manager to prepare and maintain the project plan for the performance of this Service Description which will include the activities, tasks, assignments
- ❑ **Discovery session** to help IBM teams understand client expectations, projections, and risks/concerns
- ❑ **Project Reporting** to track consolidated IBM recommendations, action items, owners, and completion
- ❑ **Support Oversight** ongoing elevated oversight of critical support cases and management visibility regarding your event/planning
- ❑ **Retrospective** post-Event to review lessons learned and better prepare you for your next Event

Next Steps Toward Readiness



1

Leverage Self-Help resources to understand & follow best-practices during your planning and preparations

2

Complete Risk Self-Assessment and discuss results and concerns with your IBM CSM

ibm.biz/OMS-EventReadiness-Qs

3

Contact your IBM CSM to discuss deeper IBM assistance through Enhanced Event Readiness engagement

4

Join us again in later this summer for a Panel Discussion with IBM Performance Experts and peak day mitigation techniques!

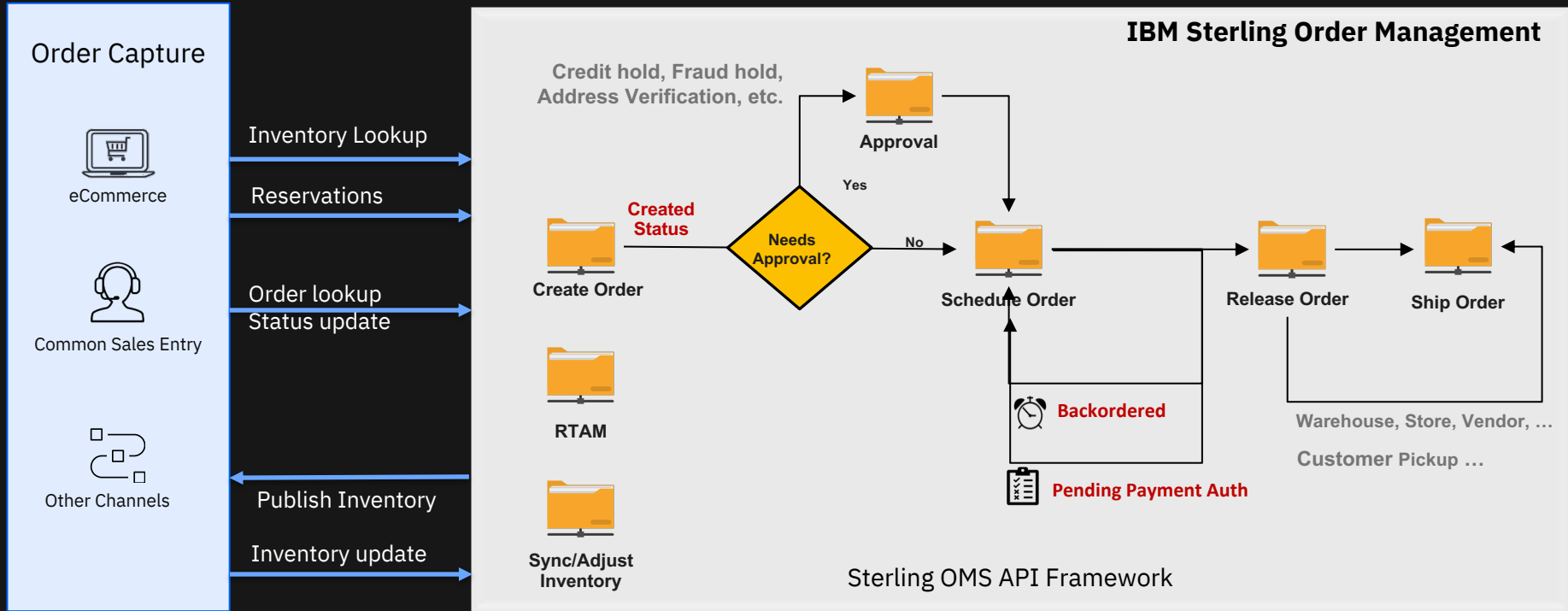
Questions?

Click **Participants** and then click **Raise hand**  next to your name.

IBM Sterling

Happy Path Order Workflow

...things happen...



Availability Lookup Considerations



If your inventory lookup is not right, then it will impact other component areas.

- *Reservations*
- *Order Capture*
- *Schedule Order*
- *Release Order*

ATP vs. Promising API

Use Optimal APIs and output template.

Procurements, optimizations (such as Cost, Date, etc.).

Shipment transfers, constraints (such as Ship Complete, Ship Single Node, etc.).

- `getATP, checkAvailability, getAvailabilityCache`
- `getAvailableInventory, findInventory`

Lead Times

Whenever we review customer's scheduling rules, we typically see that lead times are set to default (30/60 days).

How does “lead time” impact availability lookup APIs? What should be the optimal value?

Can availability lookup be kept simple as scheduling process will handle the optimization?



[Lead Time configuration](#)

Solver Optimization

There is quite a bit of optimization that happens within promising API (Cost), as such there might be certain level of performance implications.

What are mandatory and safe guarding properties that we can set to ensure cost based optimization done by promising logic does not add overhead? How to keep optimization light-weight?

`yfs.solver.MaxChoiceFailures`

Smart Sourcing

Legacy

Reading and processing inventory for nodes that do not contain inventory is costly.

To improve performance, smart sourcing can be used to dynamically determine the nodes to consider for sourcing product items. How does this really work and what benefits can we expect for inventory lookup promising APIs?



[Smart Sourcing](#)

Optimistic Lock Avoidance

Inventory locking only on low availability -

`yfs.hotsku.lockOnlyOnLowAvailability =Y`

Initial availability assumed to be high and lock is avoided until availability becomes low (an entry is present in the `INV_INVENTORY_ITEM_LOCK` table)

Availability calculation does not lock the `YFS_INVENTORY_ITEM` record unless corresponding record exists in `INV_INVENTORY_ITEM_LOCK` for the demand type.

**Not applicable for OMoC NextGen 2.0*

- **Reservations**
- **Order Capture**
- **Schedule Order**
- **Release Order**



[All about HotSKU
Optimistic Lock Avoidance](#)

INV_INVENTORY_ITEM_LOCK

- Rolling average calculation determines if availability low
- Entries for item-node-demand type combination
- If availability becomes high, then `INV_INVENTORY_ITEM_LOCK` record is removed.

Other Impacting Properties

- `yfs.hotsku.useGranularLockingForItem` Whenever availability at a ship node is low, a lock record will be inserted for Item-ship node combination
- `yfs.Hotsku.useAvailabilityAcrossNodes` Considers availability across all the nodes to decide if availability is low and item needs to be locked.

Purpose

- The purpose for the lock record. Valid values:
 - 10 (Lock as Availability is now low),
 - 11 (Use previous Hot SKU functionality).
 - Added 20 – Low availability when granular locking is enabled.

Avoid locks to `YFS_INVENTORY_ITEM`

- `yfs.hotsku.lockItemOnInventoryChanges=N`
- Inventory adjustments continue to contend on the same `YFS_INVENTORY_ITEM`
- Lock contention moved from item level down to the item-supply/item-demand level

Damage Control

Techniques to bring Stability to Order flow

Throttle the costly transactions, or make it faster.

- **Inventory Lookup**
- **Reservations**
- **Order Capture**
- **Schedule Order**
- **Release Order**



Understanding Backlog

- Understanding the pending workload and Business commitments.
- What is the priority?

JVM Profile

- Reduce # of JVMs
- Reduce # of Threads
- Review Heap setting in reference to verbose GC logs.

Segregate Workload

- Orders are not getting scheduled as there are excessive orders getting backordered due to insufficient inventory, how do we improve the scheduling rate for new orders? (OrderFilter=N)
- Releasing SFS and BOPIS orders are priority, how do I ensure those orders get released within desire SLA? (e.g. SFS vs. BOPIS vs. Ship from DC)

Find the bottleneck

- What are the dependent transactions (Payment Auth, Holds, etc.)?
- What are the costly transactions (On success events, etc.) ?
- What were the recent changes?

RTAM

Often, we have seen the issues where Activity based RTAM doesn't perform and not able to publish inventory updates fast enough.

There could be many reasons for this, but main reason that we have seen are:

1. Burst of inventory updates from source system.
2. Slow Supply correction or DG override UE's.
3. Sudden inventory activity insert due to store capacity updates.
4. Configuration (monitoring large number of nodes) .

- **Inventory Lookup**
- **Reservations**
- **Order Capture**
- **Schedule Order**
- **Release Order**



RTAM Options

- Activity Based
- Full Sync (should we be running this instead of activity based?)
- Quick Sync (Often overlooked)
- *How to decide and benchmark the RTAM mode? Any Scenario?*

Follow Best Practices

- Impact of `createInventoryActivity`.
- *What are best practices if RTAM activity based is used to manage Store Capacity (capacity free or filled event)?*
- Avoid monitoring unwanted delivery methods.
- Avoid running full sync with node level monitoring.

Controlling Workload

- RTAM/System should be tuned to process the burst efficiently
- Also needs to be controlled at the source system i.e. OMS should not receive the inventory updates in a big batch

RTAM – Housekeeping

- *Do we have unprocessed/obsolete records in `YFS_INVENTORY_ACTIVITY`/`YFS_INVENTORY_ALERTS`?*
- *What should be the ideal size `YFS_INVENTORY_ACTIVITY` table when running activity based RTAM.*
- *My `YFS_INVENTORY_ALERTS` table had 100 Million records, should we be worried?*

Scenarios worth considering ...



Real Scenarios *(Real*

To best position for success on the OMS platform, it is important to understand how your application handles various scenarios known to challenges performance or stability..

User Exits:

Pay close attention to UE average response time, and maximum response time. Often we have seen UE's resulting into:

- Long Running queries impacting DB IO and Transaction logs
- Cause server to hang with unbounded transaction boundaries (timeout's)
- Acquiring locks and hold it for prolong time impacting other transactions.

Long running UEs can negatively impact response time of a transaction.

YFS_STATISTICS_DETAIL is a good place to identify long running UEs and the response time of the API associated with the UE. Check for maximum time specifically if UE is making external call, if it's really high then it means that operations withing UE does not have timeout set.

Have a time out set for all services reaching out to third party system to avoid holding locks on an entry for long and create hung situations.

External connections not timing out leading to hung payment execution UE, subsequently it impacted the schedule transaction.

If UE is invoking any list APIs then ensure a cap of max number of records being passed. Instance where OHK was not passed to the UE and getOrderList invoked from the UE was pulling 5k records for each invocation , leading to an OOM.

What are other best practices for custom user exits?

closeManifest API:

A store associate confirm the shipment of packages through the designated carrier service via Web Store Engagement application. As part of this process closeManifest API is called on each of the ManifestKeys returned by the addAllContainersToManifest API.

During holiday period when we have multiple shipments we may need to trigger ShipOrder task multiple time instead of doing at the End Of the day to avoid performance issue

Note : Shipping packages from the user interface involves the closeManifest API call which takes considerable amount of time. To improve the performance, the closeManifest API is called asynchronously. Perform the following steps to call closeManifest API asynchronously:

Set the yfs.closemanifest.online property to N.

Configure the YDMCloseManifestAgent time triggered agent for the CLOSE_MANIFEST transaction and start the agent. You must create an agent criteria for the ship node, associate it to an agent server, and start the agent server.



✓ GOAL: Optimal APIs & Output Template

! PROBLEM: Realtime inventory calls timing out under load and occasionally JVM went OOM.

↓ OBSERVATIONS:

- `findInventory` API used for all inventory lookups.
- `findInventory` API called without maximum records attribute, and due to insufficient inventory solver optimization logic ran recursively, leading to OOM; As a result additional OOB properties were introduced.

```
yfs.yfs.solver.WarningOrExitOnIntrupt=A
yfs.yfs.solver.IntruptAfterMinutes=10
yfs.yfs.solver.IntruptOnlyForRead=true
```
- Unoptimized DG sequence.
- Using incorrect API for inventory look up purpose.
- Excessive inventory lookups from catalog browsing pages without caching

↓ RECOMMENDATIONS:

- **Use optimal APIs and output template;** if strictly checking inventory then `getAvailableInventory` or `getAvailablityCache` APIs might be a better choose compare to `findInventory`.
- **OOB UI makes different API** calls based on what is required for specific screen. Enable trace and understand input/output, leverage API javadocs.
- **Explore smart sourcing option**, with it only relevant nodes are considered for solver optimization (solution).
- **If you are using multiple DG's** in your sourcing sequence then make sure majority of calls only hit 1st or 2nd sequence for optimal performance.
- **Implement short lived availability cache** for catalog browsing pages.

✓ GOAL: Avoid Inventory Contention

! PROBLEM: Realtime inventory calls timing out under load along with notable backlog for schedule order


↓ OBSERVATIONS:

- Overhead from excessive records in `INV_INVENTORY_ITEM_LOCK` table (8 Million with purpose 20)
- Distribution Group had capacity check turned on with very high number, which caused sever contention around `YFS_RES_POOL_CONSMPTN_DTLS`,
 - Every time promising logic runs from reservation / schedule / release / create Order the capacity consumption is updated for the same node.

↓ RECOMMENDATIONS:

- **Avoid capacity checks for DG.** If you had turned on capacity check for DG/node during phased released or cannery approach then make sure to disable the capacity. Do not workaround by increasing the capacity to a large number like 1 million.
- **If emergency changes are required to sourcing setup**, then please refer to [Recommendations](#) on possible use-cases that could be impacted due to COVID-19 for safe actions.
- **Periodically review `INV_INVENTORY_ITEM_LOCK` table** to understand the inventory contention and rectify it. In case table needs to be truncated then raise a support case with IBM.

GOAL: Avoid Intrusive Store/Call Center Search

 **PROBLEM:** Frequent DB IO utilization spikes with cascading impact to other critical transaction.


OBSERVATIONS:

- Queries causing high IO were from APIs like `getShipmentList`, `getShipmentDetails` and `getOrderList` executed via store and call center interface
- Queries were bad because either they were missing shipnode condition or searched by uncommon `YFS_PERSON_INFO` attribute or attribute value using like.
- Mitigated issue by adding indexes for common `YFS_PERSON_INFO` attributes, and corrected the resource permissions for store user such that enterprise and shipnode got stamped implicitly.

RECOMMENDATIONS:

- In-store associates must be made aware of qualified search criteria when looking up the order or shipment. If there are commonly used searches then make sure querying volumes are indexed appropriately.
 - Most common is `YFS_PERSON_INFO` attributes, not all customer uses same common attributes.
- Store users must be configured and linked with team/resource group such that general attributes like ShipNode, Organization, etc are stamped implicitly. Refer to team concept in [documentation](#).

GOAL: Avoid Runaway Queries

 **PROBLEM:** Unable to take action on certain orders via Call Center.

OBSERVATIONS:

- Excessive `YFS_ORDER_HEADER` for update queries in lock-wait.
- Single connection (Application Handle) blocking most of the queries, based on javacore it was identified that the thread holding the lock was in network socket read communicating to third party system. Exception stack indicated that this was manual multiAPI call.
- It was identified that multiAPI call was executed to settled payment for 100+ orders.
- Server had to be restarted to mitigate the issue.

RECOMMENDATIONS:

- External synchronous call from UE must have timeout set.
- Avoid batch processing with synchronous calls, as a best practice setup an integration server to process such workload asynchronously.

IBM Sterling