2022 Holiday Peak Readiness
SME Panel Discussion

—

IBM Order Management

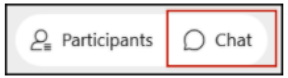
- ***Have a question?***
- ***Care to share your experiences or resolutions to similar scenarios?***

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

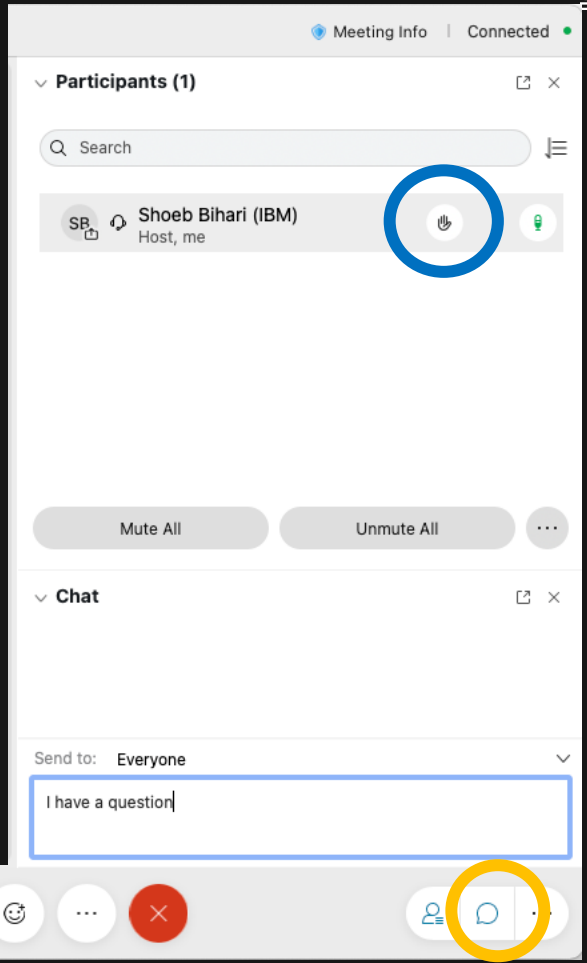
To lower your hand, click **Lower hand** next to your name in the Participants panel.

To send a chat message:

1. Open the Chat panel from the link in the lower right of the meeting window:



2. In the **Send to** or **To** drop-down list, select the recipient of the message.
3. Enter your message in the chat text box, then press **Enter** on your keyboard.



Your Holiday Readiness Team …
and today's Panel of Experts

Chris Burgess

Manager – Americas & AP Support Experience Team

Aparna Subramanian

Tech Lead / Cloud Support Architect – OMS Support

Shoeb Bihari

Technical Lead / SRE Advisor – OMS Support

Mike Callaghan

Program Director – WW Supply Chain Support

Senthil Ponnusamy

Monitoring Squad Leader - OMS Support

Amar Jyothi Annamalai

Performance Architect , Inventory Visibilty SME

Bobby Thomas

Performance Architect OMS Engineering

Vishal Arora

Advisory Software Engineer – IBM Sterling

Paresh Vinaykya

Executive Technical Account Manager, Expertise Connect

Vijay Gosvai

Developer – OMS Payments SME

IBM OMS Holiday Readiness

Mike

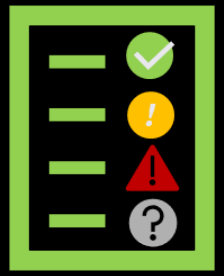
*Our IBM OMS Support mission aims to **proactively partner with you** to identify and address potential risk, while quickly mitigating any issue that does arise.*

*We have worked relentlessly to establish a **stable platform** and robust collection of proven **self-help best practices** focused on peak performance and stability.*

*Now we are focused on building a **deeper understanding and engagement** with clients in need of direct, prescriptive guidance via our Event Readiness offering.*

Best Practices – Growing Foundation

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!



In case you missed it …

2022-05-19 | Holiday Readiness_2022 KickOff
2021-07-08 | Holiday Readiness Kickoff
2021-09-16 | Peak Considerations
2020-07-09 | Peak Readiness Best Practices
2020-09-24 | Peak Day Mitigation
2020-10-20 | Best practice for smooth Peak

See also..

Recommendations for Improving Performance

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

Mike

Let’s Discuss

1. Performance tips around Payment Collection/Execution agents
2. Common reasons for YFS_ORDER_HEADER contention
3. Web Store performance and best practices
4. Inventory sync/load with GIV
5. Inventory Visibility APIs and best practices

Chris

Payment servers –
Records not being processed

As a part of regular order flow (without delayed authorization), Orders are created and authorized, followed by scheduling

Payment process seem 'stuck' despite agents running. On restart, the servers begin to process records again to hit the issue again after some time.

Other symptoms include Payment agent being very slow, internal Queue being having consistently high queue depth

What are the common causes for payment servers going stale?

- ❑ Multiple payment collection threads processing the same order
- ❑ Large number of old/pending orders picked up for processing
- ❑ Time out not implemented for calls to external gateways(paypal etc)

getJobs:

```
SELECT YFS_ORDER_HEADER.ORDER_HEADER_KEY,YFS_ORDER_HEADER.LOCKID FROM YFS_ORDER_HEADER
YFS_ORDER_HEADER WHERE ( YFS_ORDER_HEADER.PAYMENT_STATUS IN ( 'AWAIT_PAY_INFO' , 'AWAIT_AUTH'
, 'REQUESTED_AUTH' , 'REQUEST_CHARGE' , 'AUTHORIZED' , 'INVOICED' , 'PAID' , 'RELEASE_HOLD' , 'FAILED_AUTH'
, 'FAILED_CHARGE' , 'VERIFY' , 'FAILED' ) ) AND YFS_ORDER_HEADER.AUTHORIZATION_EXPIRATION_DATE <=
sysdate AND YFS_ORDER_HEADER.DRAFT_ORDER_FLAG = 'N' AND YFS_ORDER_HEADER.ENTERPRISE_KEY =
'DEFAULT' AND YFS_ORDER_HEADER.DOCUMENT_TYPE = '0001' AND NOT EXISTS ( SELECT '1' FROM
YFS_ORDER_HOLD_TYPE WHERE YFS_ORDER_HOLD_TYPE.ORDER_HEADER_KEY =
YFS_ORDER_HEADER.ORDER_HEADER_KEY AND ( YFS_ORDER_HOLD_TYPE.HOLD_TYPE IN ( 'PENDING_INV_UPDATE'
) ) AND YFS_ORDER_HOLD_TYPE.STATUS < '1300' )
```

Orders pending

Run query to identify how many orders will be picked by the payment collection agent

Check the OHKs of the records returned, to see how many old orders are still being processed

Problematic Orders

Orders with large number of charge transaction records

```
select order_header_key,
count(*) from
yfs_charge_transaction
group by order_header_key
having count(*) > 500
order by order_header_key
desc
```

Manual DB updates- can be avoided

User exit in invoked status for `OPEN` charge transaction records

Orders with invalid payment types, requiring to be pushed to future authorization date

Design Issues

Upfront authorization created with dynamic charge transactions

Unique transaction ID for calls to external gateway resulted in errors for orders having multiple settlements

Payment execution UE not handling all errors in the code.

Database Contention – YFS_ORDER_HEADER

High concurrent workload against YFS_ORDER_HEADER, as such if there are any design issues, then it could lead to performance issues due to row level locking.

```
SELECT YFS_ORDER_HEADER.* FROM YFS_ORDER_HEADER YFS_ORDER_HEADER WHERE
(YFS_ORDER_HEADER.ORDER_HEADER_KEY = '20220718092604512945 9467') FOR UPDATE

SELECT YFS_ORDER_HEADER.* FROM YFS_ORDER_HEADER YFS_ORDER_HEADER WHERE ( ( 1 = 1 ) )
ORDER BY ORDER_HEADER_KEY
```

Not sure? Check SST Database Metric dashboard for long running and lock-wait queries.

What are optimal configuration to minimize YFS_ORDER_HEADER contention?

- ❑ Validate the input before calling get APIs.
- ❑ Keep transaction boundary small when locking the record, before fetching DB object sort the items to avoid deadlock
- ❑ Use appropriate timeout when making external call, specifically if API is locking the record.
- ❑ Use pagination (`getPage`) when calling get APIs, and use appropriate `SelectMethod`, possible values are `NO_LOCK`, `NO_WAIT`, `WAIT`, etc
- ❑ Add timeouts as part of the API input :

```
<Order OrderHeaderKey="'20220718092604512945 9467"
        QueryTimeout="3" TimeoutLockedUpdates="Y"/>
```

or use global timeout:

```
yfs.agentserver.queryTimeout, yfs.ui.queryTimeout
```

Use Cases

Calls from eCom Order history / details pages

`getOrderList`, `getCompeteOderDetails`

API from UE logic

`getOrderDetails`

API calls from Store/CC UI.

`getShipmentList`, `getShipmentDetails`, `getFulfillmentSummaryDetails`

Select Method & Query Timeout's

- ❑ `NO_LOCK/NONE`, `NO_WAIT`, `WAIT`
- ❑ `QueryTimeout`, `TimeoutLockedUpdates`, `ReadUncommitted`
- ❑ Transaction boundary

Avoid Deadlocks

Blank Queries

Not having API input validation could lead to queries without any filterable predicates, resulting in severe performance issue.

Output template, certain APIs will lock the `YFS_ORDER_HEADER` row if the output template has Order element.

Use pagination to limit the data fetched/returned.

Redundant Processing of Order

Excessive `YFS_CHARGE_TRANSACTION` records causing issue for Payment Execution/Collection agents.

Hold Processing: **`omp.holdtype.reprocess.interval.delayminutes`**

Order Monitor repeatedly processing same order:

`yfs.yfs.monitor.stopprocessing.ifcondition.eval.false=Y`

Order Monitor

Web Store –
Performance and Best Practices

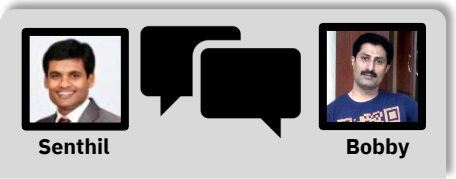
Store users might notice performance issues during pick/pack/ship process especially during store closing hours when shipment backlog gets cleaned out if there are any design issues or due to heavy shipment (and others) tables.

Common symptoms from store application,

- Slowness in store home page
- Pick / pack operations are slow
- Shipment confirmation action is taking a long time

Best practices / recommendations to improve store performance ...

- ❑ Design Optimizations
- ❑ Test Coverage
- ❑ Caching
- ❑ Database Hygiene



Design Optimizations

Trimmed API output template

Asynchronous `closeManifest` API call

Number of widgets in home screen

Polling interval for alerts

Pagination for list APIs

`GetStoreBatchList` with optimum values for `MaxNumberOfShipments`, `NoOfShipmentLinesForNewBatch`

Shipment/order search

Test Coverage

Configuration to be same as production

Test with enough data in the system

Individual API vs Full load testing

Stores are treated like DC's

Entity Caching

Enable database table caching

Review and fine tune the cache sizing based on cache hit vs miss ratio and table size

`YFS_ATTR_ALLOWED_VALUES`

`YFS_ATTR_ALLOWED_VAL_LOCALE`

`YFS_REGION`

`YFS_REGION_DETAIL`

Database Hygiene

Keep the shipment tables light

Move exceptions to closed state to be purged and keep YFS_INBOX table light

Ensure indexs are in place for related tables specially for extended columns

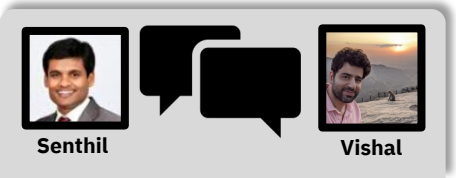
Inventory load / sync with GIV

When you reconcile the OMS inventory picture with actual inventory at the nodes, process could be running for a long time or cause an unexpected behavior if there are issues with design or configurations

- Inventory sync is taking very long time to complete
- Inventory picture updated to 0 for items not in feed
- Invalid items get created in the OMS catalog
- Inventory sync is overlapping with other critical processes

What are the recommendations to improve the inventory load performance?

- ❑ Understand the recommended design approach
- ❑ Know the common issues and how its handled through configuration properties
- ❑ Run inventory, inventory temp and audit purge to keep the related table lightweight.
- ❑ Plan the inventory load activities to timebox it



Happy Path Scenario

loadInventoryMismatch API

↓

syncLoadedInventory API
(ApplyDifferences=N & ON_INV_MISMATCH_LIST enabled)

↓

adjustInventory API

Common issues

Start *syncLoadedInventory* API after *loadInventoryMismatch*

Usage of *CompleteInventory* flag in full sync vs delta sync

ValidateItems flag in *loadInventoryMismatch* API

ON_INV_MISMATCH vs ON_INV_MISMATCH_LIST

InsertInventoryItem flag in syncLoadedInventory API

Pass *UseHotSKUFeature="Y"* in adjustInventory API

Housekeeping & Performance

Inventory purge

Inventory supply temp purge

Inventory audit purge

Timeboxing the load

When to run the inventory load?

How to avoid the overlapping with peak load / critical process?

Post inventory sync process - RTAM

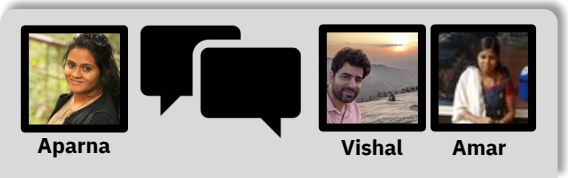
OMS-IV integration –
Increase in cancellations or BackOrders

When IV phase 2 is used, IBM Order Management reaches out to IV during order flow for –

- ❑ Availability checks
- ❑ Reservations
- ❑ Updating demand and supplies

If there is an unexpected spike in the number of cancellations or backorders, what are various factors that need to be reviewed?

- ❑ Sourcing configurations still maintained at OMS
- ❑ Capacity definition, capacity availability or cache capacity
- ❑ Node marked as having incorrect availability- Entry in YFS_INVENTORY_NODE_CONTROL
- ❑ Node notification time
- ❑ Calendar, and Carrier service schedule
- ❑ Scheduling Rule, Optimization and Line dependencies
- ❑ Unsupported delivery method



Override API input OMS-IV integration

Handle to override default input generated from integration adapter to call an IV API

Interface IMassageInputBeforeIVCall to modify input to RemoteIVAPICall.

Customer overrides property - iv_integration.input.massager=<class>

Inventory at IV

Review supply/demand updates- Use the supply and demand audit feature to review the updates.

Review node availability- Use get Detailed Node Availability API

Review DG availability-Upcoming feature includes a new rate limited API to trigger DG sync

Links

Events

Order flow through IV gives rise to events.

If not using different delivery methods, disable one of the delivery methods

Create a periodic script to pull messages from failed end points (reach out to support if not familiar with the process/end points)

Links

Token Generation

Reuse tokens , valid for 12 hours.

Excessive token generation and Calls to IV failing

Hold the token in JVM cache, reuse this across all calls in that JVM till the JVM is alive.

OMS Platform – API Enhancements

IBM recommends you review our critical API enhancements and implement the changes needed to enable performance optimization. Be sure to understand behavior and ensure no impact to order flow!

reserveAvailableInventory

- With Inventory Visibility Integration (phase 2) and later, the **ReserveAvailableInventory** API is enhanced to combine reservation calls to Inventory Visibility, whenever applicable.
  - Therefore, you must set **yfs.UseAggregatedReservationsForIV** property to **“Y”**.
  - Note: In major upgrade 22.2 this aggregation property will be enabled by default.
- The smart sourcing logic of IBM Sterling Inventory Visibility (phase 2) is disabled.

createOrder & changeOrder

- The SQL query to fetch records from **YFS_REGION_DETAIL** and **YFS_ITEM** in the **createOrder/changeOrder** API is optimized to improve performance when creating or changing large orders.

manageCapacityReservation

- The **manageCapacityReservation** API enhanced to support locking for capacity availability for Service resource pools (PS/DS), Locking happens depending on parameters passed in **yfs.properties**
- New attribute **LockCapacity** has been introduced:
  - To avoid locking for capacity availability, set **LockCapacity="N”** **manageCapacityReservation** input.
- The **manageCapacityReservation** API enhanced to update capacity during transaction commit, similar to order and inventory reservation APIs.
  - If **yfs.yfs.persitCapacityAdjustments** is set to **true**, pass **PersistCapacityAdjustments="Y“** in **manageCapacityReservation** API so that capacity is updated only when a transaction is committed

getResourcePoolCapacity

- The **getResourcePoolCapacity API** enhanced to read capacity availability **from Capacity Cache**.

Shoeb

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

Shoeb

Best Practices – Introspection & Housekeeping

To be best positioned for peak success, it is critical to TAKE ACTION NOW!
Look inward to understand your current gaps, and ensure ongoing housekeeping tasks are performed.

Introspection

- ❑ **Retrospect on your past incidents** - audit and evaluate any ad-hoc mitigatory actions/changes and drive improvements to avoid/automate them or redesign the impacted feature/function
- ❑ **Review API enhancements** and any previous outstanding IBM recommendations and implement them now.
- ❑ **Address your known design issues** and limitations as identified thru IBM engagements, or ensure mitigation in place
- ❑ **Verify and enhance external integration logic** as per the best practices (timeouts, retries, failovers, certificate management / expiry, etc.)
- ❑ **Validate performance test profile** and scenarios based on the previous peak volume/usage (KC link for OMoC 2.0)
- ❑ **Collaborate early with business users** (Store Associates, CSRs, etc) on expectation and current challenges, address them

Shoeb

Housekeeping

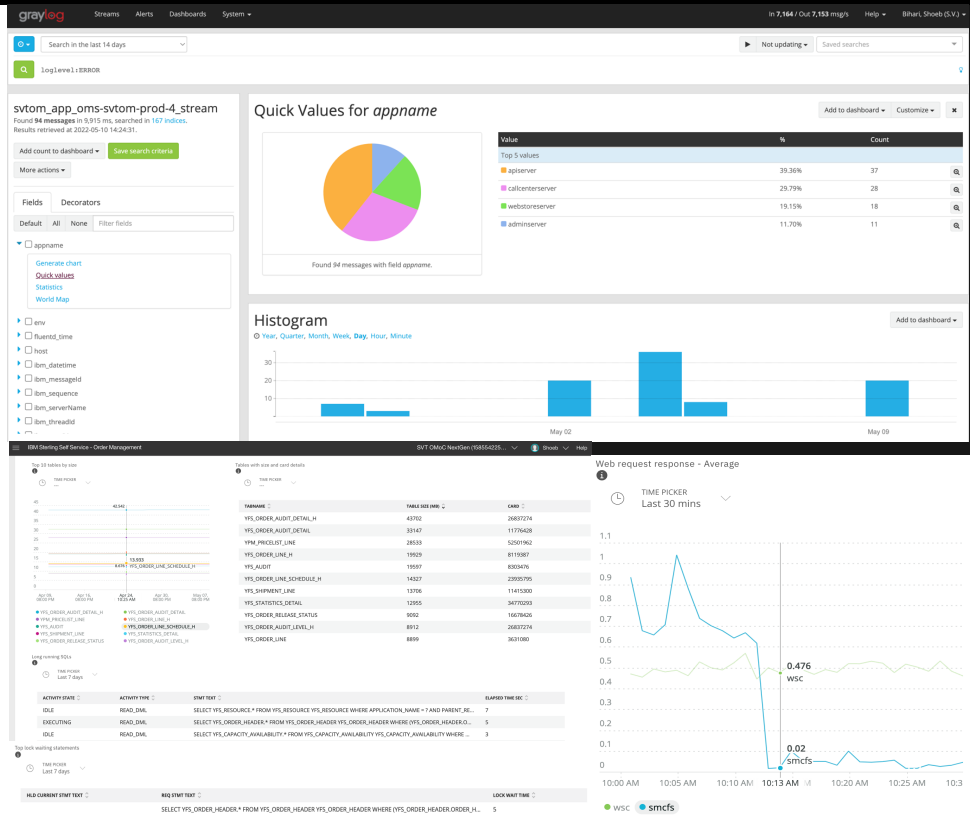
- ❑ Ensure production and preproduction master & configuration data sets are identical
- ❑ Populate preproduction transactional data by running frequent tests
- ❑ Keep the MC environment in sync and up-to date with other OMoC and toolkit environments
- ❑ Compare and ensure properties across all environments are in sync
- ❑ Neglecting/avoiding warning reported by business users could impact resolution of critical issue
- ❑ Review custom indices and schema changes, remove unused indices
  - ❑ Identify and address potential DB indices mismatch between environments.
- ❑ Participation and review proactive case to ensure proper closure

Best Practices – Self Service Tool

You have an ongoing role in being proactive! *Maintain the health and performance of your OMS applications leveraging capabilities of the Self Service Tool.*

Self-Serve Tool

- ❑ Review production performance metrics to ensure there are no anomalies or excessive errors, understand the NORMAL usage.
  - ❑ Dashboards are available to help proactively monitor application usage and performance (KC link)
  - ❑ Graylog for OMoC 2.0
- ❑ Review DB size and table growth to ensure adequate purges are enabled and running.
  - ❑ DB performance dashboards is available to help proactively monitor DB usage and performance (KC link)
  - ❑ Proactively review most expensive and long running queries both in production, load tests, and optimize them to avoid contention which will increase further under heavy peak load (rework, add index, RUNSTATS)



Shoeb

Best Practices – Proactive self-audit checklist

Proactively audit the production environment *to identify, correct known issues in config or operation across performance, database, MQ, HotSku, RTAM, purge, and external integration.*

- ❑ Validate RTAM configuration; validate activities created by node capacity changes. Review GC overhead caused by RTAM server, if its high then consider increasing the JVM profile (i.e. heap size) (link)
- ❑ Review and tune HotSKU and OLA configuration for YOUR business scenarios; check for low inventory items INV_INVENTORY_ITEM_LOCK(link)
- ❑ Review MessageBufferPutTime relative to ExecuteMessageCreated statistic from YFS_STATISTICS_DETAIL table for any slowness; ensure JMS properties in place. (link)
- ❑ Validate MQ settings, avoid long-running transactions, ensure necessary queue-depth alerts. (link), Enable retries for JMS Sender (i.e., PUT(s) message via service flow)
- ❑ Optimize long-running or expensive queries; check for lock-waits and frequent timeouts (i.e., YFC0006, YFC0009)
  - ❑ Verify the performance of YFS_PERSON_INFO queries. (link)
- ❑ Validate transactional table sizes to ensure necessary purges are being run regularly, but disabled during peak
- ❑ Review reference data cache; catch redundancy by analyzing application logs for frequent cache drops (i.e., 'Clearing cache'). Frequent refreshes of MCF reference data cache can lead to performance issues. (link)
  - ❑ Identify the use of deprecated APIs (SIP/IV)

Shoeb

- ❑ Reduce message payload by optimizing API, event templates, pull only required data.
  - ❑ Restrict output by setting the MaximumRecords in the inputs to any list API calls; use pagination (link)
- ❑ Review long running user-exit's (external calls) by reviewing maximum YFS_STATISTICS_DETAIL.STATISTIC_VALUE. Implement timeouts.
- ❑ Review order and shipment monitors for redundancy, review and remove obsolete monitor rules.
- ❑ Review agent criteria to ensure GetPendingJobs feature is used as needed. (link)
- ❑ Review errors and ensure errors are addressed to avoid noise, if not address it could mislead during crunch time, also it could cost performance during elevated load, impacts our ability to monitor the system effectively.
- ❑ Review orders for large number of charge transaction due to unforeseen payment condition and address them.
  - select order_header_key, count(*) from yfs_charge_transaction group by order_header_key having count(*) > 200 order by order_header_key desc

Best Practices – Lower Environment Health

Are your lower environments healthy? *Maintaining Dev and QA environments can prevent disruption in production, and allows you to promote best practices/configurations to the production early on*

It all starts with your lower environments!

MQ Queue Depth and Message Size

- Take action to ensure corresponding servers are healthy. Check on queues with very high queue depth, and for large message size.

Application exceptions

- Using Graylog/SST review errors and ensure errors are addressed to avoid noise, which could become misleading if new critical issue arises.
- Ensure **trace is disabled** on any API, and not left over from prior debug which will cause massive slowdowns and instability.

JVM tuning

- Select correct JVM profile (*OMoC NextGen) based on analysis from verbose GC logs or your -Xmx/-Xms parameters(Legacy)

Shoeb

Database Hygiene

- Ensure all necessary purges are running to maintain healthy & lightweight database, which in-turn minimizes performance issues
- Disable unnecessary transaction audits (Order Audits, General Audits, etc)
- Periodically review table size and cardinality of the dataset. Focus on following critical tables:
  - YFS_EXPORT/YFS_IMPORT
  - YFS_AUDIT
  - YFS_ORDER_AUDIT_DETAIL
  - YFS_ASYNC_REQ / YFS_ASYNC_REQ_ERROR
  - YFS_ORDER_RELEASE_STATUS
  - YFS_INBOX/YFS_REPROCESS_ERROR
  - YFS_INVENTORY_SUPPLY_TEMP
  - Any custom tables with CLOB/BLOB data

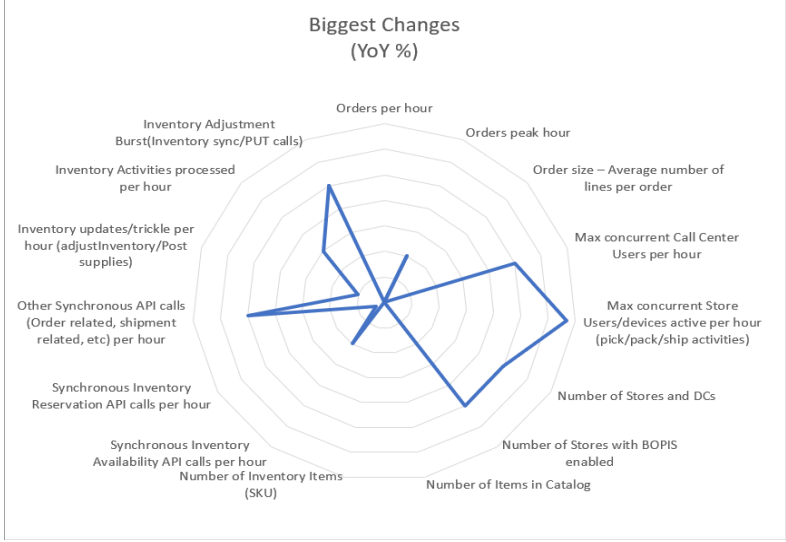
Risk Self-Assessment

IBM OMS 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

Mike



Enhanced Event Readiness

IBM Sterling experts are available through a 3-5 month comprehensive technical engagement to help proactively prepare for peak season success on the IBM Order Management solution.

First come, first serve! Contact your IBM CSM or account team for details!

Performance Health Check

A performance health check performed by an IBM Expert Labs Performance Expert, with a comprehensive technical review of your IBM OMS solution and a set of recommendation to achieve performance and stability.

- **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,
- **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)

Mike

Proactive Consultation

An ongoing proactive consultation by the OMS Support Performance Squad, with prescriptive recommendation based our vast experience and consolidated best practices in performance, stability, and peak volumes.

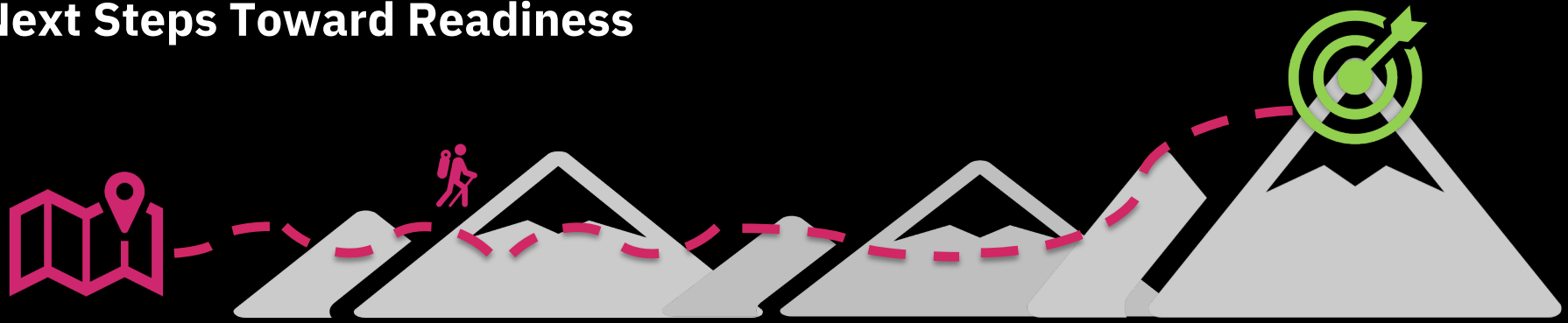
- **Technical enablement** of best practice config, proactive housekeeping, leveraging SST tooling
- **Proactive consultation** including test coverage, OMS workload prioritization, runbook preparation
- **Gap analysis** of application configuration, and validation of alignment to proven best practices
- **Risk identification** based on production operation, monitoring, utilization, projected workloads
- **Retrospective** to ensure closed loop on historical critical Support cases, production alerts
- **Stand-by Support** of SMEs informed of your solution, available during critical business periods

Support Advocacy

Ongoing high-touch cadence and communication by a named IBM Support Advocate to guide the engagement, reporting, tracking, completion of recommendations.

- **Planning** to help schedule and maintain the overall program execution and track tasks, deliverables
- **Project Reporting** to track consolidated IBM recommendations, action items, owners, and completion
- **Discovery session** to help IBM teams understand client expectations, projections, and risks/concerns
- **Ongoing elevated Support visibility** of critical cases and escalation of potential blockers to your event
- **Retrospective** post-Event to review lessons learned and better prepare you for your next Event

Next Steps Toward Readiness



1

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

ibm.biz/OMS-EventReadiness-Qs

Chris

2

Continue your journey by applying the various Holiday best practices shared through this web series, and test for expected peak workloads.

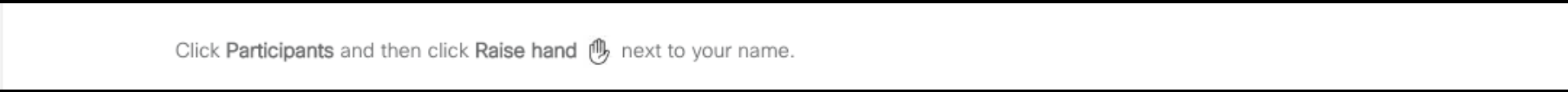
3

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

4

Join us again in September for tips on peak day execution!

Questions?



For additional information, please contact:

—

Mike Callaghan
mcallagh@ca.ibm.com