Overview of Fulfillment

Presenter's Name
Presenter's Title

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.
The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Oracle Training Materials – Usage Agreement

Use of this Site ("Site") or Materials constitutes agreement with the following terms and conditions:

1. Oracle Corporation ("Oracle") is pleased to allow its business partner ("Partner") to download and copy the information, documents, and the online training courses (collectively, "Materials") found on this Site. The use of the Materials is restricted to the non-commercial, internal training of the Partner's employees only. The Materials may not be used for training, promotion, or sales to customers or other partners or third parties.

2. All the Materials are trademarks of Oracle and are proprietary information of Oracle. Partner or other third party at no time has any right to resell, redistribute or create derivative works from the Materials.

3. Oracle disclaims any warranties or representations as to the accuracy or completeness of any Materials. Materials are provided "as is" without warranty of any kind, either express or implied, including without limitation warranties of merchantability, fitness for a particular purpose, and non-infringement.

4. Under no circumstances shall Oracle or the Oracle Authorized Delivery Partner be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this Site of Materials. As a condition of use of the Materials, Partner agrees to indemnify Oracle from and against any and all actions, claims, losses, damages, liabilities and expenses (including reasonable attorneys' fees) arising out of Partner's use of the Materials.

5. Reference materials including but not limited to those identified in the Boot Camp manifest can not be redistributed in any format without Oracle written consent.

Agenda

- Fulfillment Pipelines and Events
- Running Fulfillment Module
- Integrating with an External System

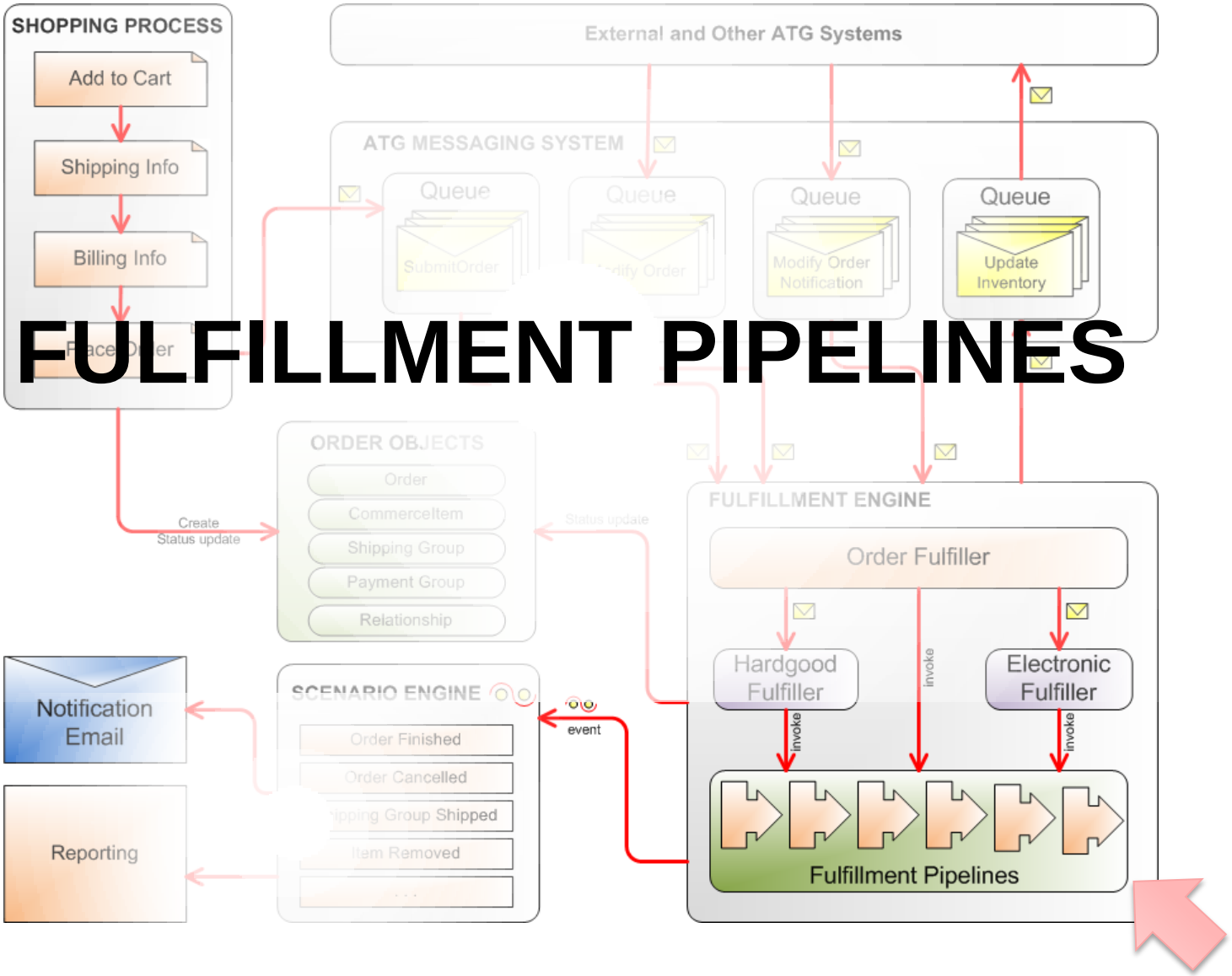
Learning Objectives

At the end of this lesson you should be able to:

- Understand the Fulfillment Pipelines
- Learn about the Fulfillment Events and Scenarios
- Run the Fulfillment Server in an ATG installation
- Integrate the fulfillment framework with external systems.

Section 1:
Fulfillment Pipelines and Events

FULFILLMENT PIPELINES



Fundamentals of Pipelines

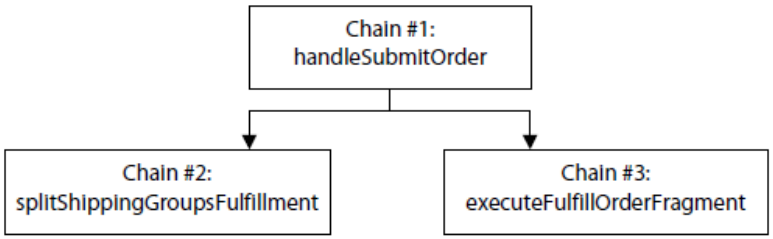
- A pipeline chain is a series of actions to be executed in a sequence.
- Each chain consists of processors, each performing a specific function.
- ATG Fulfillment has multiple pipeline chains which are executed depending on where the customer is in the purchase process.
- This mechanism allows developers to easily customize the chain by inserting their own processors.
- Pipeline chains are configured in an XML file.
- The Fulfillment PipelineManager component controls processor chains and can be used for customization.

Fulfillment Pipelines

- Pipelines play an important role in the fulfillment system.
- Fulfillment pipeline manager registers various fulfillment chains via definition file located at: /atg/commerce/fulfillment/fulfillmentpipeline.xml.
- Each of the chains runs within a transaction.
- Some examples of pipelines are:
  - **handleSubmitOrder** chain is triggered when OrderFulfiller receives a SubmitOrder message.
  - **handleModifyOrder** chain is triggered when a fulfiller receives a ModifyOrder message.
  - **handleModifyOrderNotification** chain is triggered by OrderFulfiller receiving a ModifyOrderNotification message.
  - **executeFulfillOrderFragment** chain verifies that each shipping group is in a state that is ready for fulfillment, and sends FulfillOrderFragment messages out to the appropriate fulfillers.

Example: SubmitOrder Chain

- The following series of chains is triggered when the OrderFulfiller receives a SubmitOrder message.

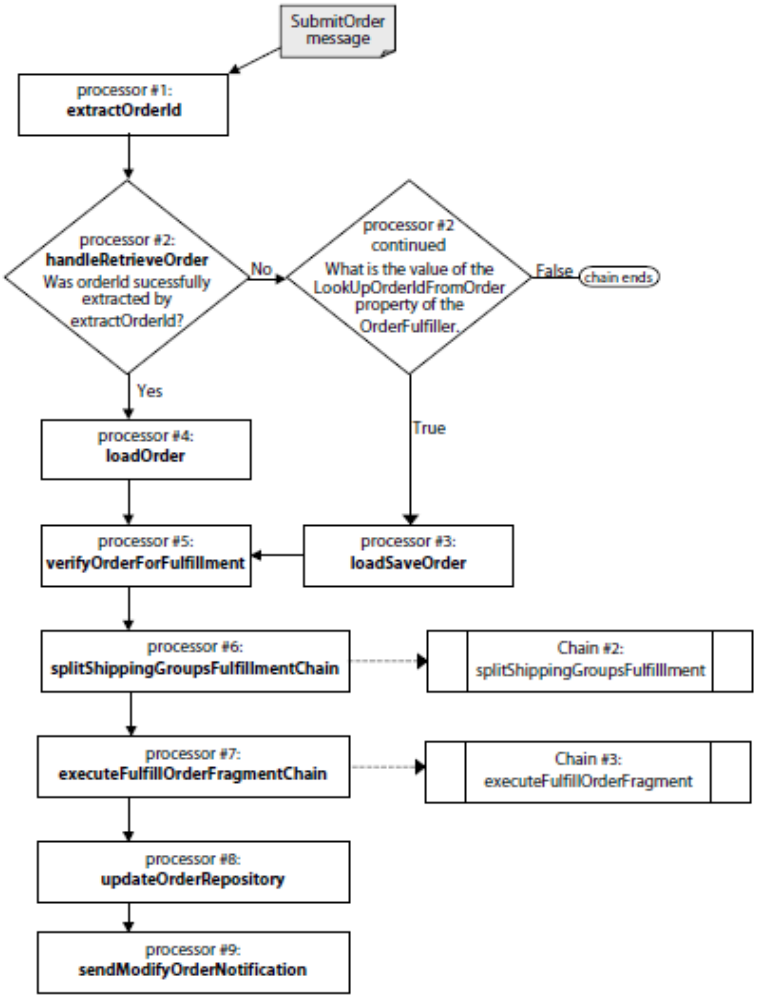


- ***Chain #1: handleSubmitOrder:***
  - This chain is triggered when OrderFulfiller receives a SubmitOrder message.
  - Next slide shows the processors involved in this chain.
  - This chain invokes splitShipingGroupsFulfillment chain and executeFulfillOrderFragment chain.

Fulfillment Pipelines

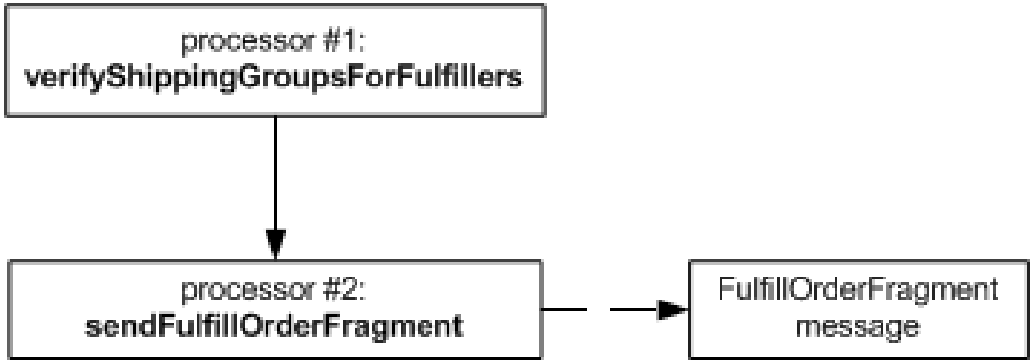
The handleSubmitOrder chain has:

- **extractOrderId**: Get the orderId from SubmitOrder message.
- **loadOrder**: Load order from order repository.
- **verifyOrderForFulfillment**: Checks if state of order is valid using OrderFulfillmentTools.
- **splitShippingGroupFulfillmentChain**: Runs splitShippingGroupFulfillment chain.
- **executeFulfillOrderFragmentChain**: Iterates through shipping groups and runs executeFulfillOrderFragment chain.
- **updateOrderRepository**: Update order in the repository to save changes.
- **sendModifyOrderNotification**: Sends ModifyOrderNotification message with the list of modifications performed during the execution of this chain using JMS.



Fulfill Order Fragment Chain

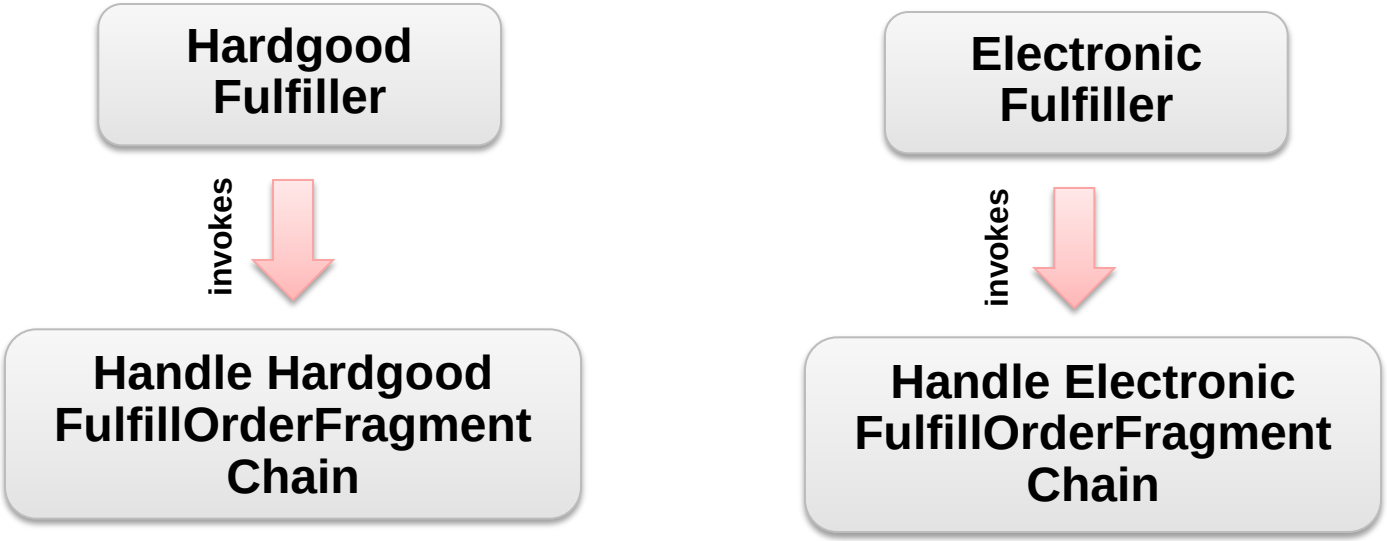
- This chain is called at the end: Submit Order Chain.
- It verifies if the shipping groups passed in the order can be handled by their fulfillers and sends FulfillOrderFragment messages to the fulfillers.
- The messages contain the shipping groups the fulfillers are responsible for.
- FulfillOrderFragment is then processed by either the ElectronicFulfiller or the HardgoodFulfiller.



Hardgood and Electronic Fulfillers

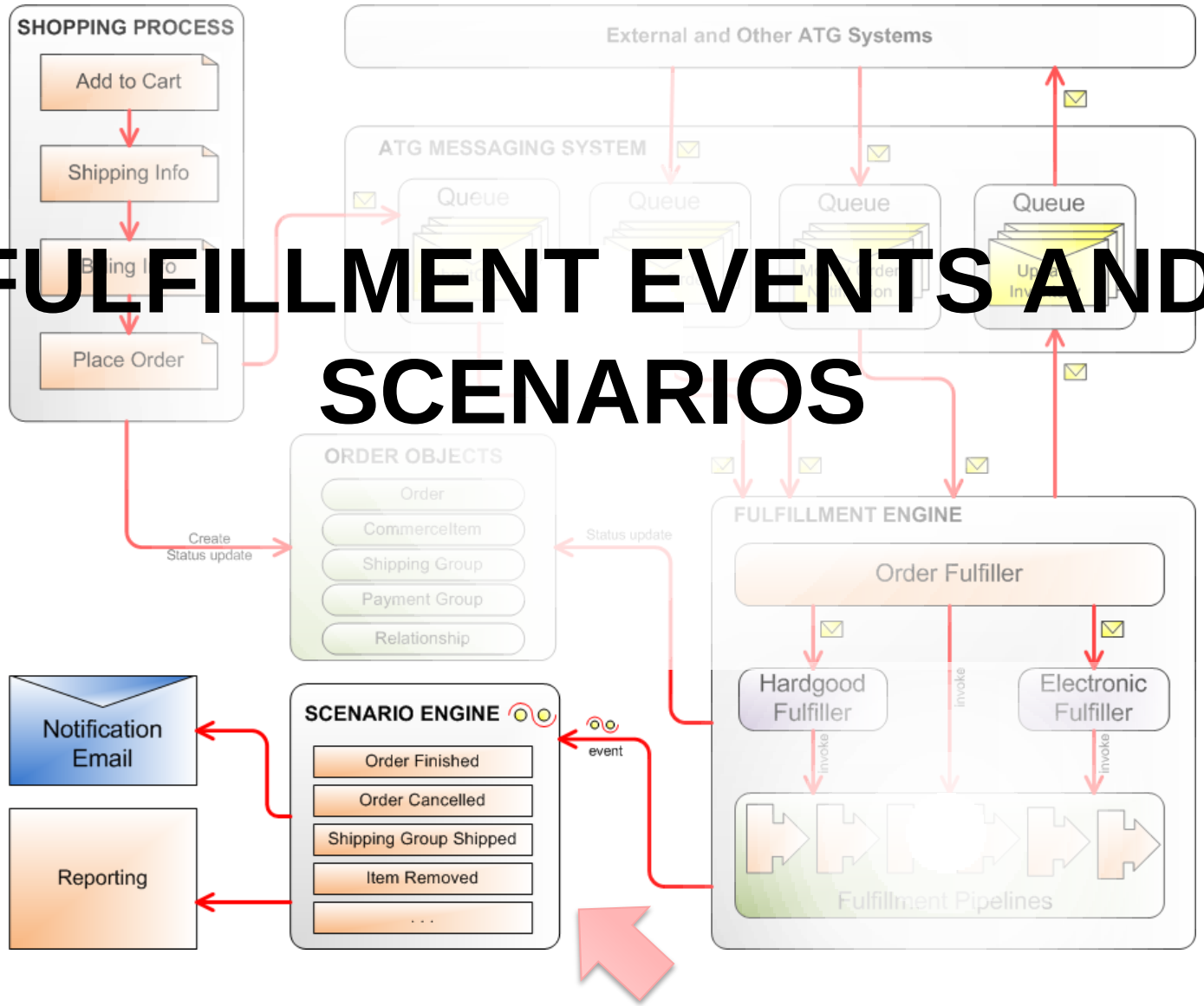
- Depending on the items in the tables and their fulfillers, HardgoodFulfiller or SoftgoodFulfiller would receive the FulfillOrderFragment.
- Mapping of fulfiller to shipping group can be found in component \atg\commerce\fulfillment\Configuration fulfillerShippingGroupClassMap.
- Similarly, the ports that each fulfiller listens to can be found in fulfillerPortNameMap in component \atg\commerce\fulfillment\Configuration.
  - This configuration component can be extended to add more properties for fulfillment to be referred to by custom components.
  - Thought should be given when adding more properties as typically these properties are common to all fulfillers.

Hardgood and Electronic Fulfillers



- HardgoodFulfiller executes handleHardgoodFulfill OrderFragment chain.
- SoftgoodFulfiller executes handleElectronicFulfill OrderFragment chain.

FULFILLMENT EVENTS AND SCENARIOS



Order Fulfillment Events

- Order Fulfillment events are created and sent by the OrderChangeHandler in the order fulfillment system.
- The OrderChangeHandler listens to the various ModifyOrderNotifications being delivered in the system and constructs one of three basic events:
  - **OrderModified,**
  - **ShippingGroupModified,**
  - **PaymentGroupModified.**
- Each event has a subtype that gives more detail on the reason.
- Note that these events are different than notification messages that the fulfillment server uses.
- You can as an example deliver a promotion to the user when OrderModified/FINISHED message is received.

Scenarios in the Fulfillment Process

- The fulfillment process uses the scenario engine to provide features to customers.
- The scenario engine listens for the order fulfillment event messages and can trigger appropriate scenarios.
- The OOTB scenarios included with ATG Commerce:
  - **OrderFinished** sends an email when order is finished.
  - **ShippingGroupShipped** sends an email when something ships.
  - **UnavailableItems** notifies the customer of unavailable items.
  - **OrderCancelled** sends an email when order is cancelled.
  - **ItemRemoved** sends an email when item is removed from order.
  - **PaymentGroupChanged** sends an email when payment group changes.
  - **SubmitOrder** notifies the customer that order has been submitted.

Section 1 ✔
Check Your Understanding

Name a few OOTB fulfillment scenarios in ATG Commerce.

Answer:

OrderFinished, ShippingGroupShipped, OrderCancelled, etc.

Section 1 
Check Your Understanding

What does the OrderCancelled scenario do in ATG Commerce?

Answer:

It notifies the customer that the order is cancelled.

Section 1 
Check Your Understanding

What are the three basic events the fulfillment system publishes?

Answer:

OrderModified, ShippingGroupModified, and PaymentGroupModified.

Section 1 ✔
Check Your Understanding

Which pipeline chain is executed by the hardgood fulfiller?

Answer:

handleHardgoodFulfill OrderFragment chain.

Section 1 
Check Your Understanding

What chains does the handleSubmitOrder chain invoke?

Answer:

splitShippingGroupsFulfillment and executeFulfillOrderFragment chains.

Summary

- ATG fulfillment has multiple pipeline chains which are executed depending on where the customer is in the purchase process.
- Some of the pipeline chains are handleSubmitOrder, handleModifyOrder,handleModifyOrderNotification, executeFulfillOrderFragment, etc.
- Order fulfillment events are created and sent by the OrderChangeHandler in the order fulfillment system.
- In ATG Commerce scenarios are provided that take action when a fulfillment event is received.

Section 2:
Running Fulfillment Module

Running the Fulfillment Server

- Fulfillment module should be assembled with your application to use the fulfillment server.
- Fulfillment module could be deployed and run as a separate server to get easier maintenance.
- Only one of the instances of the site application should include the fulfillment module, otherwise you could get state conflict issues.
- You can use the default fulfillment module or extend it to build up your own fulfillment server.

Locking in Fulfillment

- You should run a ClientLockManager when running a fulfillment server.
- This ensures that no component handles more than one message per order at any given time.
- All fulfillment components use the lock manager located at /atg/dynamo/service/ClientLockManager.
- Every ATG Commerce component should use the same ClientLockManager.
- The lock acquired is for the key that is returned by the method getKeyForMessage in OrderFulfiller and HardgoodFulfiller. The default implementation returns the orderId specified in the message.

Example: Using Locking in Fulfillment

```
TransactionDemarcation td = new TransactionDemarcation();
try {
  td.begin(getTransactionManager(), td.REQUIRED);

  getClientLockManager().acquireWriteLock(pOrderId);
  LockReleaser lr = new LockReleaser(getClientLockManager(),
    getTransactionManager().getTransaction());

  lr.addWriteLock(pOrderId);

     <insert your code here>

  catch (DeadlockException de) {
    if(isLoggingError())
      logError(de);
    return false;
}
...
```

Fulfillment Server Fault Tolerance

- ATG Commerce fulfillment framework uses SQL JMS messages so you do not need a complex system of redundant fulfillment servers with automatic failover.
- Fulfillment work occurs within the context of a transaction.
  - If the fulfillment server goes down, all current transactions roll back.
  - The message is resent after a transaction rolls back because message delivery and processing occur within the same transaction.
  - Any messages that are sent to the fulfillment server while it is down, including those that are resent, are persistent in the database and will be delivered once the fulfillment server is back online.

Replacing the Default Fulfillment System

- You can replace the fulfillment system that ships with ATG Commerce with another fulfillment system.
- For example, if you wanted to use a test fulfillment system with or in place of the existing fulfillment system. Follow these steps:
  - Configure your new fulfillment system within Patchbay to subscribe to the sqldms://Fulfillment/SubmitOrder topic.
  - If your test fulfillment system is using a separate repository for orders, configure a new OrderManager with a new OrderRepository.
  - If you want the scenario server to perform actions based on fulfillment events, configure Patchbay so that your fulfillment systems sends the events and the ScenarioManager listens for them.

Section 2 ✔
Check Your Understanding

What is the recommended number of fulfillment engines that need to be run in production?

Answer:

Only one instance should be run.

Section 2 
Check Your Understanding

What is the role of ClientLockManager?

Answer:

It ensures that no component handles more than one message per order at any given time.

Section 2 
Check Your Understanding

How do you ensure redundancy when running fulfillment engine?

Answer:

ATG Commerce fulfillment framework uses SQL JMS. So, only once instance is needed.

Section 2 ✔
Check Your Understanding

What does the default implementation of getKeyForMessage in OrderFulfiller return?

Answer:

It returns OrderId.

Section 2 
Check Your Understanding

What is the behavior of the fulfillment engine when it goes down?

Answer:

All messages are rolled back and resent when the server comes up.

Summary

- **Fulfillment module** should be assembled with your application to use the fulfillment server.
- **Only one** of the instances of the site application should include the fulfillment module, otherwise you could get state conflict issues.
- You should run a **ClientLockManager** when running a fulfillment server.
- ATG Commerce fulfillment framework uses SQL JMS messages. So you do **not** need a complex system of redundant fulfillment servers with automatic failover.
- The default fulfillment system can be replaced if needed.

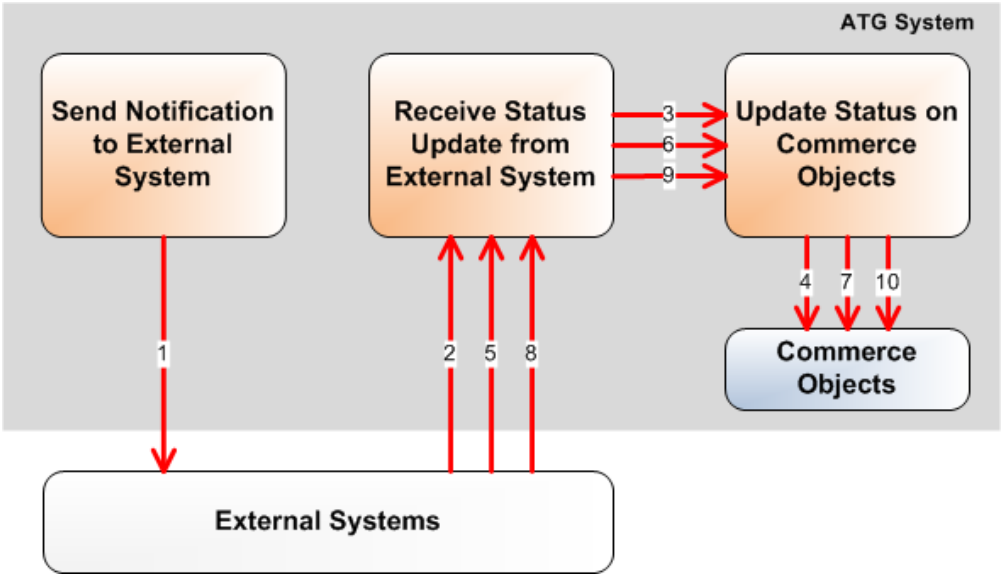
Section 3:
Integrating with an External System

Integrating with an External System

- The order fulfillment framework can be integrated with an external system that can act on the order. An example is an external order management system or shipping system.
- There are three major activities that need to be performed:
  - Send a notification to the external system to act on the order or its subcomponent.
  - Receive a status update from the external system.
  - Act on the notification and update status on the commerce objects.
- The first two activities are asynchronous. There is possibly a period of time between sending the notification to the external system and receiving an update.

Interactions with External System

- ATG sends a notification to the external system (1).
- The external systems sends multiple data elements back (2, 5, and 8).



- Internally, the recipient of the message calls on another module (3, 6, and 9) to update the status of commerce objects (4, 7, and 10).

Approaches to Sending a Notification to External System

- The first part of integration is to figure out when to notify the external system.
- There are three approaches you can use to write your code for integration
  - Insert a new **pipeline processor** in the appropriate chain. This is the recommended approach.
  - Write a **JMS message sink and message source**. Register for ModifyOrderNotification. Your code can be invoked when the appropriate commerce object is changed.
  - Write a **scheduler** that queries the order repository for commerce objects in a particular state. Your code can then be invoked and deal with the objects. This has performance implications and is also not real time.

Choice of Transportation of Message

- Transportation deals with the mechanism by which your message is delivered to the external system.
- Philosophically, this choice should be and typically is made by the external system.
- Some of the choices are:
  - Method invocation on a provided library (e.g. Cybersource).
  - Drop it in the Enterprise Service Bus (ESB) (e.g. Tibco).
  - REST/HTTP/RMI based invocation on external server or external system invoke REST based service on ATG System.
  - JMS Message drop off on external system or pick up from exposed queue or topic in ATG. This is not as decoupled.
  - FTP drop off to a remote folder location or FTP pickup from a local location by external system.

Approaches to Receiving a Notification from the External System

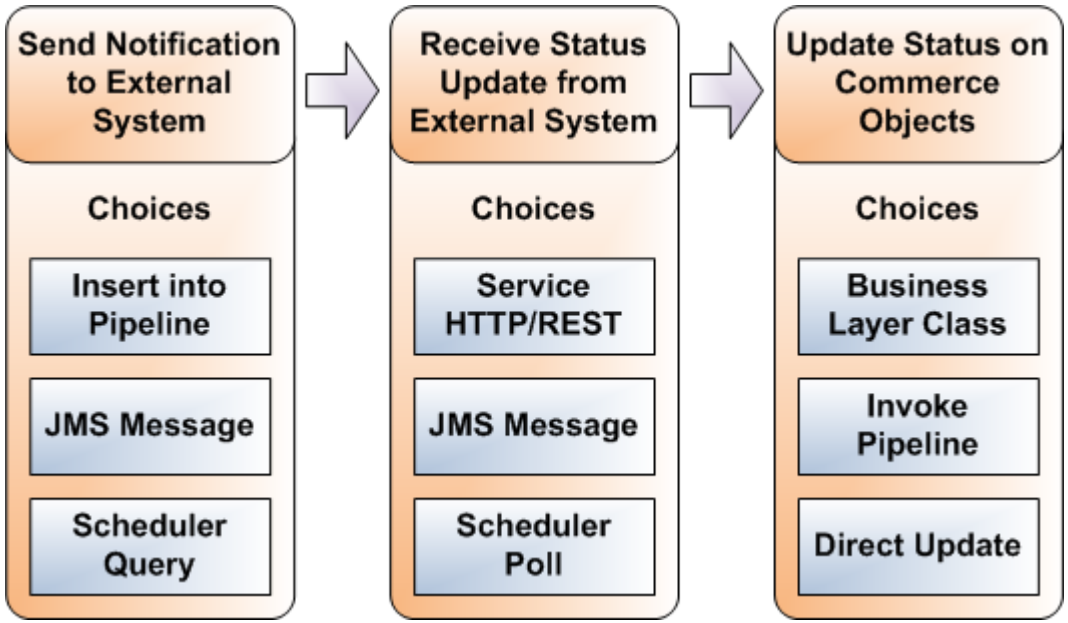
- If there is a change of state to a relevant object in external system (such as shipment has shipped), the external system can notify you.
- There are three methods to getting notified:
  - Expose a **service** via HTTP, REST, or RMI to receive update messages. This is the **recommended** approach.
  - External systems could connect to ATG via **JMS** and send ModifyOrder message.
  - You can write a **scheduler** to **poll** the external systems for a status update on interested objects.
- The method you pick depends on the capabilities of the external system to some degree.

Approaches to Updating the Status of Commerce Objects

- Once you receive a notification, you will typically update the status of the commerce objects in this process.
- There are three approaches to updating status in the order of recommendation:
  - Use the high level **business class** (such as HardgoodFulfiller) to invoke a public method provided for that purpose.
  - Invoke the relevant **pipeline chain** that serves that purpose. Or,
  - **Directly update** the status of the commerce object and send out a ModifyOrderNotification message.
- The particular method you pick may depend on the business requirement.

Summary of Integration Choices

- You could pick any combination of choices to integrate with external system.
- For example, for shipping: You can insert a pipeline to notify the shipper, poll for status, and use a business layer class to update the commerce objects.
- The method you pick for each of the three will depend on the business requirements and other constraints of the external system.



INTEGRATING WITH AN EXTERNAL SHIPMENT SYSTEM

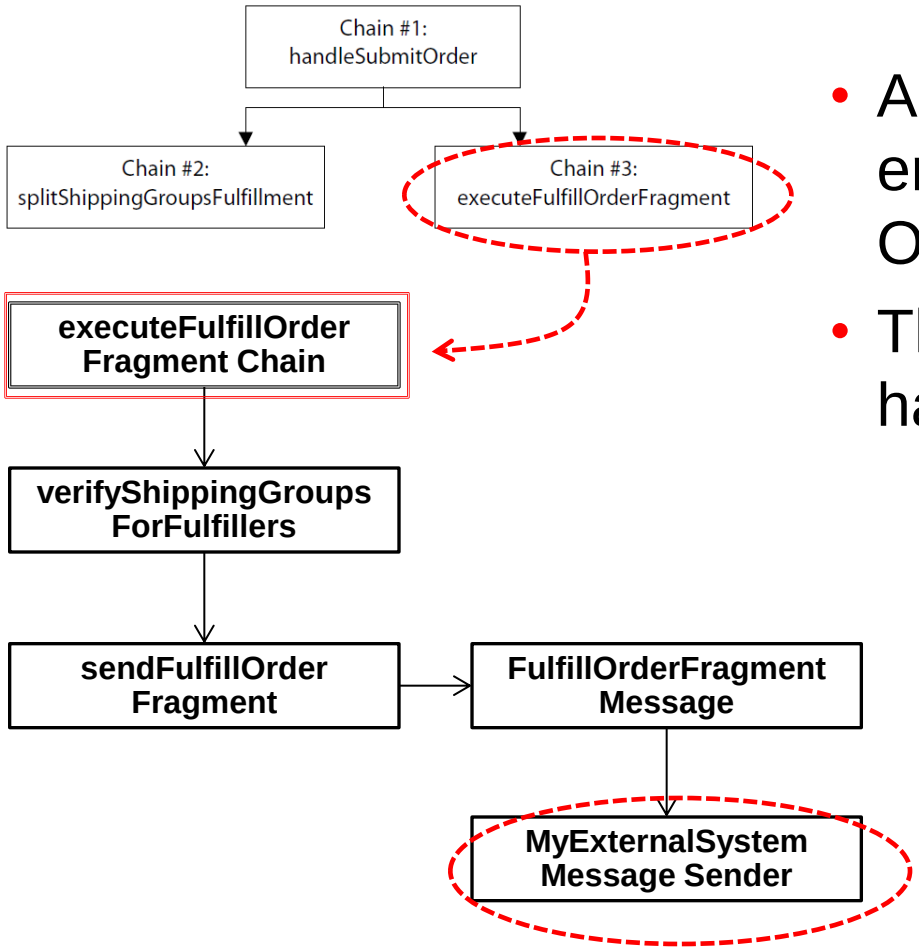
Integrating with an External Shipping System

- The order fulfillment framework can be integrated with an external shipping system that actually ships the order to the customer
- By default, there is no integration with an external shipping system.
- For this integration, we will:
  - Use the pipeline to detect when to send a message to the external system. Write a processor and insert into the pipeline component.
  - Use a REST based service to receive notification from the shipper.
  - On receipt of a message, use the HardgoodFulfiller to notify of the update.

Integration levels

- Typical integrating with external shipping system to submit an order can happen on various levels.
  - Submit **entire order** and all its shipping groups (hardgood and softgood) to the shipping system.
  - Submit **hardgood & softgood shipping** to two different shipping systems.
- To submit entire order, developer could add a new processor at the end in executeFulfillOrderFragment chain.
- To submit by shipping group, developer could add a new processor at the end in handleHardgood FulfillOrder Fragment & handleElectronic FulfillOrder fragment chain.

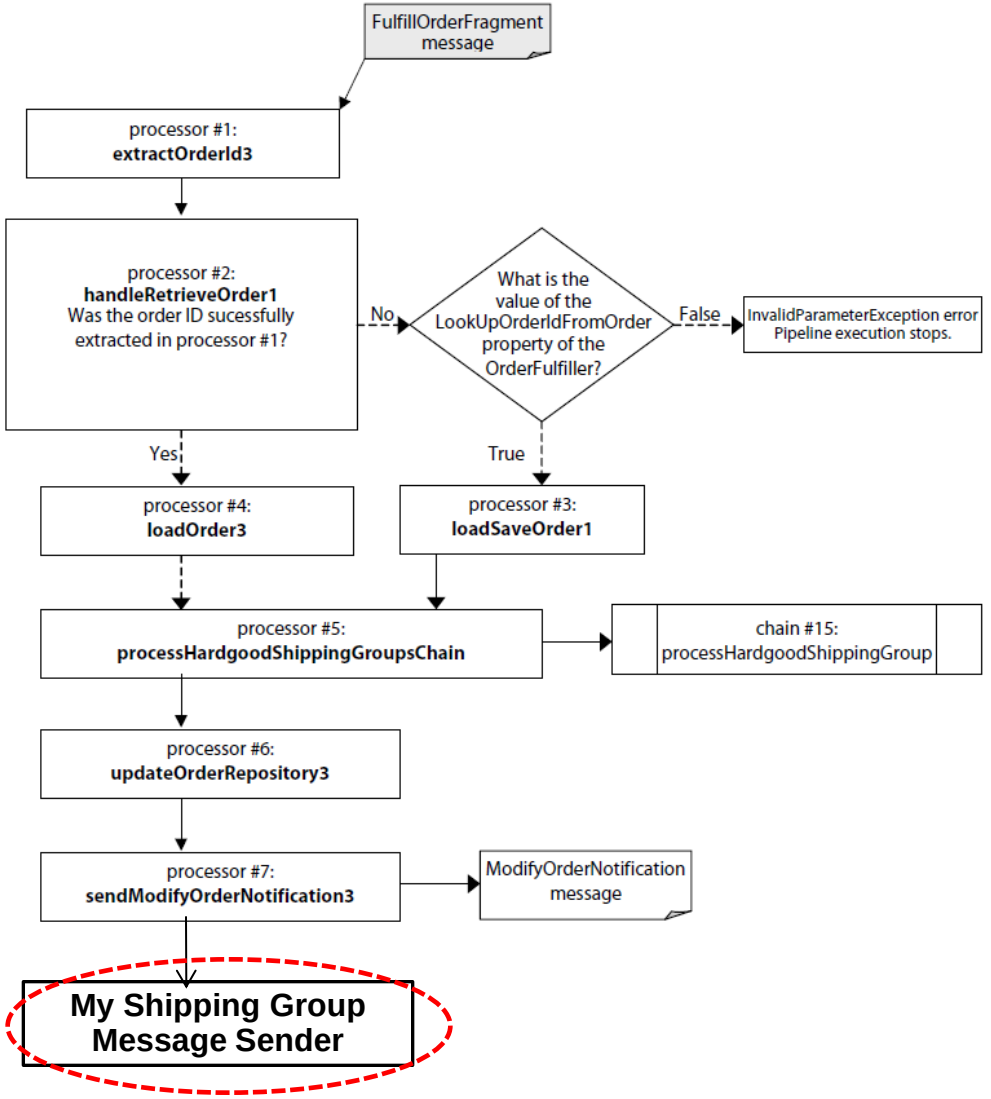
Submitting the Entire Order



- Add a new processor at the end of executeFulfill OrderFrament Chain.
- This chain is called by the handleSubmitOrder chain.

Submitting at the Shipping Group Level

- The picture shows the processors executed when FulfillOrder fragment message is received.
- Add a new processor at the location shown.



Receiving Updates from the External System

- When a notification is received from an external system, the developer can indicate to the order fulfillment system that the shipping group has shipped using:
  - Call HardgoodFulfller.shippingGroupHasShipped() method. This is the recommended approach.
  - If the update is not for shipped state, developer can find a pipeline such as ShippingGroupHasShipped pipeline and invoke the pipeline.
  - If no pipeline is available for the type of update, developer can change the commerce object (shipping group relationship) to the appropriate state and send a modify order notification.

Section 3
Check Your Understanding

What are the three questions that need to be answered to integrate the fulfillment system to an external system?

Answer:

When to send a message, how to receive a notification, and what to do with the notification data.

Section 3

Check Your Understanding

What are some of the transportation choices for delivering a message to external system?

Answer:

Connecting via REST based services is popular. FTP based feed pickup and drop off was popular and is finally giving way to more modern methods. Use of ESB or other middleware is common in large companies.

Section 3 
Check Your Understanding

What are the various methods of receiving notification from external systems?

Answer:

REST based service, JMS messages, and polling. If an FTP feed is used, you could use a scheduler to poll for new feed files.

Section 3
Check Your Understanding

What is the recommended approach to gain control for sending a message while integrating with an external shipping system?

Answer:

Pipeline.

Section 3
Check Your Understanding

Which pipeline can be used to gain control for sending a message while integrating with an external shipping system?

Answer:

executeFulfillOrderFragment if entire order is going to external system or handleHardgood FulfillOrder fragment chain if only one shipping group is being sent to an external system.

Summary

- The order fulfillment framework can be integrated with an external system that can act on the order.
- To achieve this you need to know when to send a message, how to receive a message, and how to update the commerce item.
- You can use pipeline process, JMS message, or scheduler to gain control when an external message needs to be sent.
- You can expose a REST call, receive a JMS message, or poll the external system to be notified when an update is available on an external system.
- You can use the business layer, pipeline process, or direct update method to modify the state of the commerce object.
- To integrate to an external shipment system, you could use a pipeline to gain control when a message needs to be sent, expose a REST based service to receive updates, and use the OrderFulfiller class to make updates to the commerce objects.

Q&A

ORACLE IS THE INFORMATION COMPANY


ORACLE®