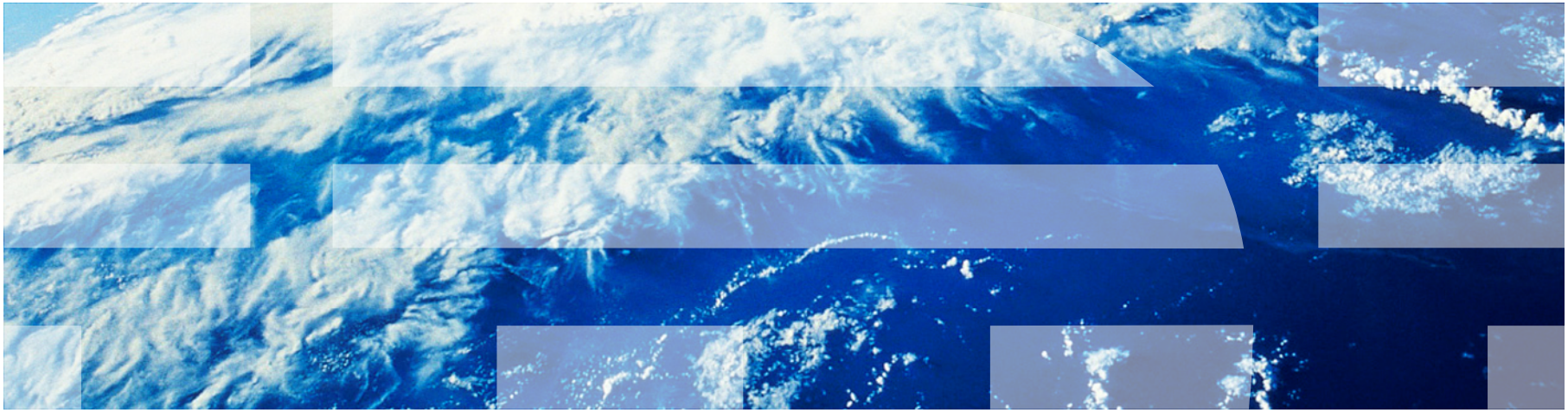
FEP 8 REST API Enhancements Technical Overview

IBM WebSphere Commerce V7



REST API Enhancements in FEP 8

- As of FEP 7:
  - REST APIs for basic B2C flows (since FEP 4)
  - Search REST APIs (since FEP 7)
  - REST framework with BOD mapping and command/data bean mapping support (since FEP 7)
  - REST API interoperability (since FEP 7)
- New in FEP 8:
  - REST APIs for all Aurora store B2C and B2B flows
  - RESTification of Aurora store
  - Search REST API enhancements
  - Configuration-based command/data bean mapping
  - REST API interoperability enhancements
  - REST security enhancements
  - Additional REST framework enhancements

New REST API Methods (Existing Resources)

Resource	API Methods
associated_promotion	GET store/{storeId}/associated_promotion?q=byName - finds an associated promotion by its name
espot	GET store/{storeId}/espot/{name} - updated to return default content title and marketing content with image map
person	GET store/{storeId}/person/@self/optOuts - returns the opt outs of the current user GET store/{storeId}/person/{userId} - find a person by its ID GET store/{storeId}/person?q=usersICanAdmin - finds persons the current user can admin POST store/{storeId}/person/{userId}?action=updateMemberGroup - updates the member groups of the user
cart	GET store/{storeId}/cart/{orderId}/usable_ship_charges_by_ship_mode - returns the usable shipping charges of a cart GET store/{storeId}/cart/@self/buyer_purchase_order/{buyerPurchaseOrderId} - finds a buyer purchase order by its ID GET store/{storeId}/cart/@self/usable_billing_address/{orderId} - returns the usable billing addresses of a cart GET store/{storeId}/cart/@self/pattribute/{initKey_referenceNumber} - returns a payment attribute by its ID GET store/{storeId}/cart/@self/payment_policy_list - returns the payment policies POST store/{storeId}/cart/{orderId}/set_pending_order - sets a cart as the current pending order POST store/{storeId}/cart/create_order - creates a cart POST store/{storeId}/cart/copy_Order - copies a cart POST store/{storeId}/cart/{orderId}/order_Calculate - recalculates a cart PUT store/{storeId}/cart/@self/delete_order_item - deletes an order item DELETE store/{storeId}/cart/{orderId}/cancel_order - cancels a cart

New REST API Methods (Existing Resources) (cont'd)

Resource	API Methods
order	GET store/{storeId}/order?q=findByParentOrderId - finds an order by its parent order ID GET store/{storeId}/order?q=findByStatusExt - finds orders by status GET store/{storeId}/order?q=findChildOrderByOrderItemId - finds a child order by order item ID GET store/{storeId}/order?q=findConfigurationByOrderItemId - finds a configuration by order item ID GET store/{storeId}/order?q=findQuoteByExternalQuoteId - finds a quote by its ID GET store/{storeId}/order?q=findQuoteByStatus - finds quotes by status GET store/{storeId}/order?q=findScheduledOrder - finds scheduled orders

New REST API Methods (New Resources)

Resource	API Methods
aggregate	GET store/{storeId}/aggregate?q={queryName} - finds an aggregate
api	GET api - returns the resources on this server GET api/aggregated - returns the resources on the WC and Search servers GET api/resource/{resourceName} - describes the APIs of a resource
approval_status	GET store/{storeId}/approval_status/{approvalStatusId} - finds an approval status by its ID GET store/{storeId}/approval_status?q=all - finds all approval statuses for the current user GET store/{storeId}/approval_status?q=buyerApprovals - finds buyer approval statuses for the current user GET store/{storeId}/approval_status?q=orderApprovals - finds order approval statuses for the current user POST store/{storeId}/approval_status/{approvalStatusId}?action=updateApprovalStatus - updates an approval status
contract	GET store/{storeId}/contract?q=byPaymentTermConditionId - find a contract by its payment term condition ID
configuration	GET store/{storeId}/configuration/{configurationId} - finds a configuration by its ID GET store/{storeId}/configuration?q=all - finds all configurations GET store/{storeId}/configuration?q=byConfigurationIds - finds configurations by their IDs

New REST API Methods (New Resources) (cont'd)

Resource	API Methods
organization	GET store/{storeId}/organization?q=organizationsICanAdmin - finds organizations the current user can admin GET store/{storeId}/organization?q=rolesICanAssignInOrg - returns roles the current user can assign in an organization POST store/{storeId}/organization/buyer - registers a buyer organization and its admin
member_group	GET store/{storeId}/member_group/{memberGroupId} - find a member group by its ID GET store/{storeId}/member_group?q=explicitlyIncludedOrExcluded - find member groups that explicitly include or exclude a user GET store/{storeId}/member_group?q=manageable - find member groups manageable by the current user
page_design	GET store/{storeId}/page_design?q=byLayoutActivityId - finds a page design by layout activity ID GET store/{storeId}/page_design?q=byLayoutId - finds a page design by layout ID GET store/{storeId}/page_design?q=byObjectIdentifier - finds a page design by object identifier
page	GET store/{storeId}/page/{pageId} - finds a page by its ID GET store/{storeId}/page/name/{name} - finds a page by its name GET store/{storeId}/page?q=byCatalogEntryIds - finds pages by catalog entry IDs GET store/{storeId}/page?q=byCategoryIds - finds pages by category IDs GET store/{storeId}/page?q=byNames - finds pages by their names GET store/{storeId}/page?q=byPageIds - finds pages by their IDs GET store/{storeId}/page?q=byUrlConfigurable - find pages with or without configurable URLs

New REST API Methods (New Resources) (cont'd)

Resource	API Methods
widget_definition	GET store/{storeId}/widget_definition/{widgetDefinitionId} - finds a widget definition by its ID GET store/{storeId}/widget_definition/identifier/{identifier} - finds a widget definition by its identifier GET store/{storeId}/widget_definition?q=byIdentifiers - finds widget definitions by their identifiers GET store/{storeId}/widget_definition?q=byWidgetDefinitionIds - finds widget definitions by their IDs
display_price	GET store/{storeId}/display_price?q=byCatalogEntryIdsAndPriceRuleId - finds display prices by catalog entry IDs and price rule ID GET store/{storeId}/display_price?q=byCatalogEntryIdsAndPriceRuleName - finds display prices by catalog entry IDs and price rule name GET store/{storeId}/display_price?q=byPartNumbersAndPriceRuleId - finds display prices by part numbers and price rule ID GET store/{storeId}/display_price?q=byPartNumbersAndPriceRuleName - finds display prices by part numbers and price rule name
requisition_list	GET store/{storeId}/requisition_list/{requisitionListId} - finds a requisition list by its ID GET store/{storeId}/requisition_list?q=self - finds requisition lists created by the current user GET store/{storeId}/requisition_list?q=usable - finds requisition lists usable by the current user POST store/{storeId}/requisition_list - creates or copies a requisition list POST store/{storeId}/requisition_list/{requisitionListId}?action=updateItem - updates a requisition list item PUT store/{storeId}/requisition_list/{requisitionListId} - updates a requisition list DELETE store/{storeId}/requisition_list/{requisitionListId} - deletes a requisition list

New REST API Methods (New Resources) (cont'd)

Resource	API Methods
search_term_association	GET store/{storeId}/search_term_association?q=byAssociationType - finds search term associations by association type
country	GET store/{storeId}/country/country_state_list - finds countries and their states GET store/{storeId}/country/country_state_name - returns the display name of a country or state
subscription	GET store/{storeId}/subscription/{subscriptionId} - finds a subscription by its ID GET store/{storeId}/subscription?q=byBuyerIdAndSubscriptionType - finds subscriptions by buyer ID and subscription type GET store/{storeId}/subscription?q=bySubscriptionIds - finds subscriptions by their IDs
file_upload_job	GET store/{storeId}/file_upload_job/{fileUploadJobId} - finds a file upload job by its ID GET store/{storeId}/file_upload_job?q=byUploadType - finds file upload jobs by upload type GET store/{storeId}/file_upload_job?q=byMemberIdAndNumberOfDaysAndUploadType - finds file upload jobs by member ID, number of days and upload type

http://www-01.ibm.com/support/knowledgecenter/SSZLC2_7.0.0/com.ibm.commerce.starterstores.doc/concepts/cwvrestapi_fep8.htm?lang=en

RESTification of Aurora Store

- In FEP 8, we have updated the Aurora store and site widget JSPs to consume REST APIs (via the wcf:rest tag) instead of BOD services and data beans

```
<wcf:rest var="getPageResponse" url="store/{storeId}/page/name/{name}">
    <wcf:var name="storeId" value="${storeId}"/>
    <wcf:var name="name" value="HomePage"/>
    <wcf:param name="langId" value="${langId}"/>
    <wcf:param name="profileName" value="IBM_Store_Details"/>
</wcf:rest>
```

http://www-01.ibm.com/support/knowledgecenter/SSZLC2_7.0.0/com.ibm.commerce.component-services.doc/refs/rwvwcfresttag.htm?lang=en

REST Struts Actions

- In FEP8, we have also extended the WC Struts framework to support mapping Struts actions to REST APIs
- e.g. struts-config-order-rest-services.xml:

```
<action parameter="orderlist.addOrderItem" path="/AjaxRESTOrderItemAdd"
          type="com.ibm.commerce.struts.AjaxRESTAction">
          <set-property property="authenticate" value="0:0"/>
          <set-property property="https" value="0:1"/>
</action>

<action parameter="orderlist.addOrderItem" path="/RESTOrderItemAdd"
          type="com.ibm.commerce.struts.RESTAction">
          <set-property property="authenticate" value="0:0"/>
          <set-property property="https" value="0:1"/>
</action>
```

REST Struts Actions (cont'd)

- rest-template-config.xml:

```
<_config:rest-action-config>
      <resource path="{serverHost}/wcs/resources/store/{storeId}/cart" name="orderlist">
            ...
            <method name="addOrderItem" httpMethod="POST" path="">
                  <template>
                        <![CDATA[{
                              "orderId" : "$orderId",
                              "orderItem" : [
                                    {
                                          "comment" : "$comment",
                                          "productId" : "$catEntryId",
                                          "partNumber" : "$partNumber",
                                          "quantity" : "$quantity",
                                          "UOM" : "$UOM",
                                          "contractId" : "$contractId",
                                          "calculationUsage" : "$calculationUsage",
                                          "fulfillmentCenterId" : "$fulfillmentCenterId",
                                          "fulfillmentCenterName" : "$fulfillmentCenterName"
                                    }
                              ],
                              "x_calculationUsage" : "$calculationUsage",
                              "x_calculateOrder" : "$calculateOrder",
                              "x_inventoryValidation" : "$inventoryValidation"
                        }]]>
                  </template>
            </method>
```

Configuration-based Command/Data Bean Mapping

- Starting with FEP 7, we support mapping REST APIs to commands and data beans:
  - http://www-01.ibm.com/support/knowledgecenter/SSZLC2_7.0.0/com.ibm.commerce.webservices.doc/tasks/twvrestsamplecmd.htm
  - http://www-01.ibm.com/support/knowledgecenter/SSZLC2_7.0.0/com.ibm.commerce.webservices.doc/tasks/twvrestsampledb.htm
- In FEP 8, we have enhanced this feature to support configuration-based mapping, which is the ability to map REST API methods to commands/data beans via mapping XML files and with minimal coding:

```
public Response findManageable(
                    @PathParam(PARAMETER_STORE_ID) String storeId) {

          String profileName = getParameterValue(PARAMETER_PROFILE_NAME,
                               PROFILE_NAME_IBM_STORE_SUMMARY, false);
          String responseFormat = getParameterValue(PARAMETER_RESPONSE_FORMAT,
                               null, false);

          Response response = executeConfigBasedBeanWithContext(
                               MemberGroupListDataBean.class.getName(), profileName,
                               responseFormat, null);

          return response;

}
```

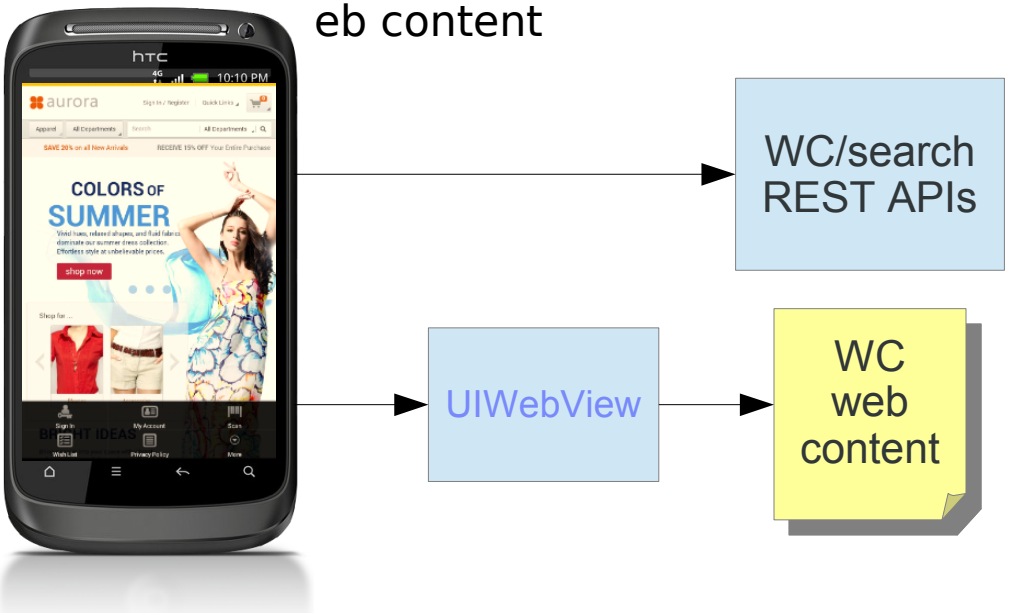
Configuration-based Command/Data Bean Mapping (cont'd)

- Mapping XML file:

```
<bean>
  <profiles>
    <profile name="IBM_Store_Summary">
      <inputs>
        <input methodName="setQueryName" inputName="q" />
        <input methodName="setUserId" inputName="userId" />
        <input methodName="setExclude" inputName="exclude" />
        <input methodName="setOrderBys" inputName="orderBy" />
        <input methodName="setPageNumber" inputName="pageNumber" />
        <input methodName="setPageSize" inputName="pageSize" />
        <input methodName="setTypeNames" inputName="typeName" />
      </inputs>
      <outputs>
        <output methodName="isRecordSetCompleteIndicator" outputName="recordSetCompleteIndicator" />
        <output methodName="getRecordSetCount" outputName="recordSetCount" />
        <output methodName="getRecordSetStartNumber" outputName="recordSetStartNumber" />
        <output methodName="getRecordSetTotal" outputName="recordSetTotal" />
        <output methodName="getMemberGroupDataBeans" outputName="resultList">
          <output methodName="getDescription" outputName="description" />
          <output methodName="getMbrGrpIdInEJBType" outputName="memberGroupId" />
          <output methodName="getMbrGrpName" outputName="name" />
          <output methodName="getOwnerId" outputName="ownerId" />
        </output>
      </outputs>
    </profile>
  </profiles>
</bean>
```

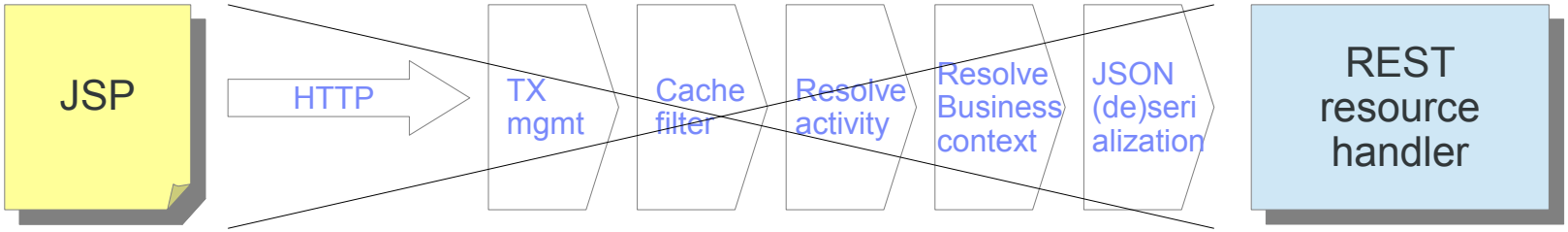
REST API Interoperability Enhancements

- Starting with FEP 7, our REST APIs can join a WC web session by consuming the WC session cookies in place of the WC REST tokens
- In FEP 8, our REST APIs can also create/update WC session cookies by specifying updateCookies=true
  - e.g. POST /wcs/resources/store/10001/guestidentity?updateCookies=true
- This will simplify the creation of hybrid apps with both native code calling our REST APIs an eb content



Local Binding

- In FEP 8, we have enhanced the wcf:rest tag to support local binding, which is the ability to call locally deployed REST APIs over Java rather than HTTP
- This eliminates overheads from HTTP connection, JSON (de)serialization, authentication etc. when calling locally deployed REST APIs
- Local binding is supported by the wcf:rest tag and the REST Struts actions
- Important note: the FEP8 Aurora starter store **requires** local binding (see assumptions)



Access Control

	Local Binding	Remote Binding
BOD mapping	Access control is: • Implemented by the BOD service • Handled by the access manager	Access control is: • Implemented by the BOD service • Handled by the access manager
Command mapping	Access control is: • Implemented by the command • Handled by the access manager	Access control is: • Implemented by the command • Handled by the access manager
Data bean mapping: • Implements the Delegator interface • getDelegate() returns a Protectable instance	Access control is: • Implemented by the data bean • Handled by the access manager	Access control is: • Implemented by the data bean • Handled by the access manager
Data bean mapping: • Implements the Delegator interface • getDelegate() returns null	Access control is: • Not handled by the access manager • Assumed to be intrinsic to the query	Access control is: • Not handled by the access manager • Assumed to be intrinsic to the query
Data bean mapping: • Does not implement the Delegator interface	Access control is: • Not implemented by the data bean • **Assumed to be implemented by the view**	Access control is: • Not implemented by the data bean • **Assumed to be site resource by the REST framework** • Handled by the access manager

REST Security Enhancements

- XSS protection
- Whitelist data validation
- SSL acceleration support
- HTTP basic access authentication support

Additional REST Framework Enhancements

- JAX-RS stack upgrade (to Apache Wink 1.4.0)
- BOD mapping enhancements

Troubleshooting

- Trace components:
  - REST framework:
    - com.ibm.commerce.foundation.rest.*
  - Individual resource handlers:
    - com.ibm.commerce.rest.<component>.handler
  - wcf:rest tag (including local binding):
    - com.ibm.commerce.foundation.internal.client.taglib
    - com.ibm.commerce.foundation.internal.client.util
  - REST Struts actions (including local binding):
    - com.ibm.commerce.struts
    - com.ibm.commerce.foundation.internal.client.util
- Find a good REST client
  - e.g. https://chrome.google.com/webstore/search/REST%20client?utm_source=chrome-ntp-icon

Assumptions

- The FEP8 Aurora starter store requires local binding:
  - It depends on site resources that would require site admin credentials in remote binding
  - The web container has only a single connection pool and therefore remote binding will reduce the number of connections available for store requests and potentially cause deadlocks
- Session affinity is required (due to use of BCS)
- ~~Search REST API method only supports JSON as response format~~
- ~~REST API interoperability requires an existing WC web session~~

Thank You!

Backup

BOD Mapping Enhancements

- The result list name can be configured using the optional URLParameterGroup/@listName attribute
- The result list can be flattened using the optional URLParameterGroup/@flattened attribute
- The data type of a URL parameter can be preserved using the optional URLParameter/@preserveDataType attribute
- A URL parameter can use another URL parameter group to map its child elements using the optional URLParameter/@type attribute
- The URL parameter type "UserData" can now handle a list of elements with Name and Value as child elements