Edge Integration via API

Sisukord

Terminology / definitions

WMS – Warehouse Management System in this document specifically K.Motion Warehouse Edge (further referred as WMS).

ERP - Enterprise Resource Planning or Accounting Application or Host System.

Download – The transfer of information from the Host/Accounting/ERP system to K.Motion Warehouse Edge WMS.

Upload – The transfer of information from the K.Motion Warehouse Edge WMS to the host accounting/ERP system.

Data Exchange Records/Transactions

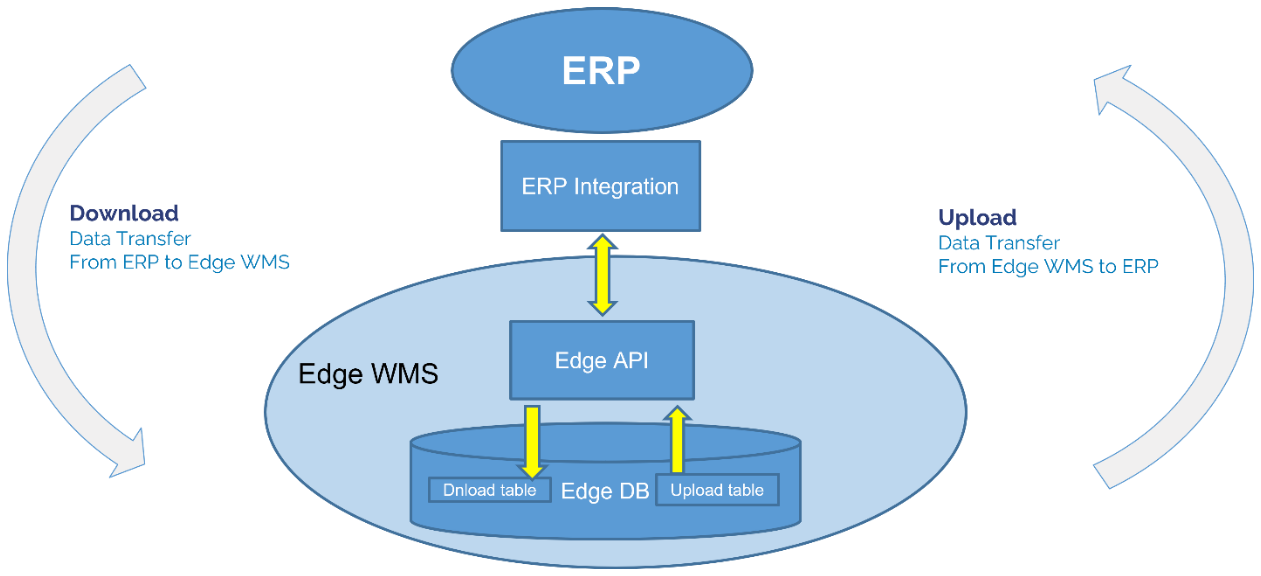
- Pick Header (PH) Record – Sales Order (SO) Header record. The data that is downloaded from the ERP to WMS
- Pick Detail (PD) Record – Sales Order (SO) Details (Order Lines) record. The data that is downloaded from the ERP to WMS
- Transfer Confirmation (XC) Record – The data that is uploaded from WMS to ERP to notify ERP about Sales Order successful download, rejected download, upload, Sales Order status change in WMS.
- Pick Confirmation (PC) Record – The data that is uploaded from the WMS to the ERP system once a sales order has been completely picked and shipped. Besides picking quantities and attributes (like serial or lot numbers), the record contains any shipping information that has been obtained for the order. Usually one Pick Confirmation record is created for one Pick Detail record.

Detailed description of all Data Exchange Records/Transactions can be found in K.Motion Warehouse Edge ERP Integration User Guide.

Integration Workflows

All WMS data exchange is processed internally in WMS via dnload and upload tables in WMS DB. These tables are used by integration for data transfer and can be accessed for read and write via WMS API. Once data is inserted in dnload table via API, WMS processes this data and populates appropriate operational tables in WMS DB. Once WMS processes (e.g. Sales Order), processing results data is posted into upload table. Integration can retrieve data from upload table via API.

Below picture illustrates data processing workflows.



Host Interchange Transactions / Data Exchange List Addressed in this document

- Sales Order Downloads (PH/PD) – Sales Order from ERP to WMS
- Sales Order Picking Confirmations (PC) – Sales Order picking confirmation from WMS to ERP
- Transaction Confirmations (XC) – Sales Order status from WMS to ERP

Full list of possible Sales Order transactions

Transactions
Download New Sales Order
Update Existing Sales Order
Delete Existing Sales Order
Lock Sales Order in ERP
Unlock Sales Order in ERP
Sales Order Downloaded
Sales Order Rejected
Sales Order Uploaded
Sales Order Picking Confirmation Upload
Processed records in Upload table deletion

Integration Scope

Current document details following ERP / WMS Transactions:

a. Sales Order Download from ERP to WMS
b. Sales Order Picking Confirmation Upload from WMS to ERP

Data structures and field mapping

- Dnload table – PD + PH: PHPD_example.txt <- embedded file (each FIELDxxx is 250 char long, UdfContainer, BusinessObjectName and UdfRowExists should always present as is shown in example)
- Upload table – UPLOAD_example.txt <- embedded file
- Pickhead, pickdetl and pickloc tables – pickhead and pickdetail are WMS Sales Order operational tables populated with SO Header and Details respectively based on data in dnload table during download process. Pickloc table is populated during download and updated during allocation and contains allocation data details. May have several records for one detail record.
- Field Mapping Tables

Download – PH/PD Records Table

Note: Logistika Pluss (LOGIS) customer code is used as an example. Should be replaced by your own customer code in Logistika Pluss's WMS here.

Sample data	Integration field	Description	WMS Edge field	Field Name (Number) in Dnload table	WMS Edge SQL
PH				FIELD001	
LOGIS-10018681		Customer's SO number will be prefixed by "CUSTOMER-" and used as Edge SO nr	5 Order Number (36C) [M]	FIELD005	PICKHEAD.PACKSLIP
LOGIS		Customer code in LP WMS	41 Client Name (15C),	FIELD041	PICKHEAD.CLIENTNAME
Logistika Pluss		Ship to Name	4 Customer Number (16C)	FIELD004	PICKHEAD.CUST_NUM
Punane 73		Omniva's parcel machine name	13 Ship Address 1 (35C) [R] + 14 Ship Address 2 (35C) [R]	FIELD013 FIELD014	PICKHEAD.SHIP_ADD1 + PICKHEAD.SHIP_ADD2
JJEELLL22040110018681		Omniva track and trace code	103 Extra3 (40C)	FIELD103	PICKHEAD.Extra3
Andri Piik		Customer's first name + Customer's last name	12 Ship Name (35C) [R],	FIELD012	PICKHEAD.SHIP_NAME
13619		Postal index	17 Ship Postal/Zip Code (10C) [R],	FIELD017	PICKHEAD.SHIP_ZIP
EE		Country	18 Ship Country (40C) [R],	FIELD018	PICKHEAD.SHIP_CNTRY
+3725154962		Customer's mobile nr, incl +country code	20 Ship Telephone Number (25C) [R]	FIELD020	PICKHEAD.SHIP_TELNO
			68 Shipment Message (20C)	FIELD068	PICKHEAD. SHIP_HD_MSG

10018681		Customer's SO number	101 Extra1 (40C)	FIELD101	PICKHEAD.EXTRA1
PD				FIELD001	
LOGIS-10018681		Customer's SO number will be prefixed by "CUSTOMER-" and used as Edge SO nr	5 Order Number (36C) [M]	FIELD005	PICKDETL.PACKSLIP
Product123		Product code	12 Product Code (100C) [M]	FIELD012	PICKDETL.PRODUCT
1		*Always 1 for one-line Sales Orders*	10 Order Sequence Number (20C) [M]	FIELD010	PICKDETL.ORDER_SEQ
Same as Order quantity		Order quantity	18 Quantity Ordered (18,6N)	FIELD018	PICKDETL.QTY_ORIGNL
1		Order quantity	19 Quantity to Pick (18,6N) [M]	FIELD019	PICKDETL.QTY_TOPICK
P		*Logistika Pluss's warehouse Id*	3 Location (10C)	FIELD003	PICKDETL.LOCATION

Upload – PC Records Table

Sample data	Integration field	Description	WMS Edge field	WMS Edge SQL
PC			FIELD001	
LOGIS-10018681		Edge SO Number	FIELD004	
10018681		Customer SO number	FIELD089	
1234567890		Serial Number (IMEI code)	FIELD020	
LOGIS		Clientname – 3PL customer name	FIELD056	

Upload – XC Records Table

Sample data	field	Description	WMS Edge field	WMS Edge SQL
XC			FIELD001	
LOGIS-10018681		Edge SO Number	FIELD003	
10018681		Customer's SO number	FIELD015	
DOWNLOAD or UPLOADED		Action	FIELD009	
LOGIS		Clientname – 3PL customer name	FIELD014	

Data preparation/formatting requirements

All data is sent via transactions as character fields types:

- Character – as is
- Integer – as charcter data
- Decimal – as charcter data with decimal point (point not comma!)
- Date – in yyyy-MM-dd format

Error Handling

API functions do not check data validity and only check API call structure. Data validation is happening during internal WMS download process when WMS operational tables are populated based on data inserted in dnload table. Response on successful data processing is posted into upload table in a form of XC records. Message DOWNLOAD means internal operational tables have been successfully populated and Sales Order is downloaded. Integration should check for this record – this is confirmation that data has been successfully downloaded. Detailed recommendation on implementing Error Handling during SO download is included in **SO download from ERP to WMS process** section below.

Processed historical integration data removal/deletion

Due to the fact that all integration data exchange happens via dnload and upload tables data recycling (processed records deletion) in these table should be ensured.

WMS handles dnload table data recycling itself. Read and processed data records are deleted. Read records, containing wrong data are marked by ’B’ and not deleted. These records are kept in dnload table for wrong data investigation and should be deleted manually.

WMS does not handle upload table data recycling. This is integration responsibility to delete processed records in upload table. If integration is not deleting processed records this may lead to data overflow and WMS abnormal behaviour. Upload table data recycling during SO upload is included in **SO upload from WMS to ERP process** section below.

Data processing flows

SO download from ERP to WMS process

Integration calls API function (DNLOADCollection – Download Sales Order) to insert records into dnload table. PD record is inserted first, PH record is inserted after all PD records for current order has been inserted. Only basic call structure of API call is validated by API at this point. Data validation happens at next step.

WMS reads SO data from dnload table and validates: mandatory fileds, data formats and other data related information.

Successfully read and validated records are used to populate WMS internal Sales Orders DB tables (pickhead, pickdetl and picklock tables). After that successfully read and validated records in dnload table are deleted and corresponding XC record with message „DOWNLOAD“ is inserted into upload table. This XC record validates to ERP integration that SO has been successfully downloaded into WMS.

Records containing wrong or missing data are marked with letter B in DNLOADED field.

Integration should check upload table XC records with message „DOWNLOAD“ to validate successful SO download into WMS. (UPLOADCollection – Sales Order Download Confirmation)

Edge's internal SO number and customer's SO number note:

In Edge, every 3PL customer has its unique 5 character customer code (like LOGIS for Logistika Pluss).

To simplify the integration effort it is proposed to use "CUSTOMER" prefix for Edge internal SO numbers. (Example: Customer's SO number=123, Edge internal SO number=CUSTOMER-123). This will eliminate the need to call Edge API function to generate Edge internal new unique SO number.

Customer SO number uniqueness note:

Customer side integration should ensure that there is no repeating SO numbers over time.

SO upload from WMS to ERP process – standard.

Integration can retrieve SO picking/shipping confirmation records from upload table.

Integration calls API function (UPLOADCollection – Sales Order Upload Confirmation) to get the list of successfully uploaded Sales Orders from upload table based on corresponding XC records in upload table with message „UPLOADED“. Edge SO numbers will be returned with CUSTOMER prefix in XC-FIELD003. Full filter for this call is included in API Calls description section below.

Example: above function returns SO-s with numbers CUSTOMER-123, CUSTOMER-124, CUSTOMER-125 – this means 3 SO's has been uploaded. Then you call below function 3 times to retrieve 3 shipment confirmations for above orders.

For every Sales Order number from above created list Integration calls API function (UPLOADCollection – Sales Order Pick Confirmation) to get specific Sales Order picking / shipping confirmation PC record from upload table. Edge SO numbers (with Customer prefix) retrieved from XC records should be used as parameter in PC-FIELD004.

Once PC and corresponding XC records for given Sales Order has been successfully processed by integration, integration has to delete appropriate XC and PC records. This will ensure that there is no upload table overflow over time.

SO upload from WMS to ERP process – simplified

Important Note: Below process will only work for one line orders with qty=1

If the Sales Orders have only one line and qty=1, a simplified approach can be used. Instead of reading XC records, PC records should be read.

Integration calls API function (UPLOADCollection – **Get Sales Order Pick Confirmation**) to get the list of sucessfully uploaded customer Sales Orders along with Serial Numbers picked and all other corresponding information from upload table based on corresponding PC records. Edge SO numbers will be returned with customer prefix in XC-FIELD004. Full filter for this call is included in API Calls description section below. Records being selected should have field upload.uploaded='0'. This will exclude duplicate upload.

All successfully processed PC records should be marked as read. This is done by updating corresponding PC record and setting field upload.uploaded='1'. Integration calls API function (UPLOADCollection – **Set Sales Order Pick Confirmation**) to make this update.

Required API calls descriptions

Logistika Pluss (LOGIS) customer code is used as an example. Should replaced by your own customer code in Logistika Pluss's WMS here.

Logon

Postman example	Request type	URL
Logon	POST	{{BaseUrl}}:30000/odata/v10/LogOn

Request body (JSON):

```
{
    "applicationId": "HighJump One Platform",
    "tenant": "",
    "userLogOnName": "",
    "userPassword": "",
    "connectionType": "integration"
}
```

Response (JSON): save two fields SessionId and SerializedAuthenticationTicket

```
"SessionId": "6542eb3d-e672-4f50-96aa-da008cfc2a08",
"SerializedAuthenticationTicket": "%3c%3fxml+version%3d%221.0%22+………",
```

<u>This ticket is valid only for one minute, therefore you should logon, do necessary API calls and then logoff.</u>

Important Note: After Logon is done, all following requests should include header "AuthenticationTicket" with value of SerializedAuthenticationTicket from Logon step

Logoff

Postman example	Request type	URL
Logoff	POST	{{BaseUrl}}:30000/odata/v10/LogOffRequest

Query parameters: sessionId=guid'22545b09-3255-4c68-9844-668015d188ab'

Query example: {{BaseUrl}}:30000/odata/v10/LogOffRequest?sessionId=guid'22545b09-3255-4c68-9844-668015d188ab'

<u>SessionId is what you get from Logon request</u>

Download Sales Order

Description: Downloads Sales Order from Integration to WMS.

Postman example	Request type	URL
DownloadSalesOrder	POST	{{BaseUrl}}:12520/odata/DNLOADCollection

Request body (JSON):

<- embedded file with examples for PD and PH record, should be sent one record per request, PD first, PH last

Response (JSON): return inserted in DB object <- embedded file (see mapping section for fields values)

Sales Order Download Confirmation

Description: Returns confirmation on successful Sales Order download from Integration to WMS.**Postman example**	**Request type**	**URL**
Sales Order Download Confirmation	GET	{{BaseUrl}}:12520/odata/UPLOADCollection

Query parameters: $filter=FIELD001 eq 'XC' and FIELD009 eq 'DOWNLOAD' and FIELD014 eq 'LOGIS'
If you are not familiar with this filtering syntax visit odata documentation section 5.1.1.1.11 Logical Operator Examples

Query example: {{BaseUrl}}:12520/odata/UPLOADCollection?$filter=FIELD001 eq 'XC' and FIELD009 eq 'DOWNLOAD' and FIELD014 eq 'LOGIS'

FIELD001 is type of record and it should always be XC
FIELD009 is type of status and should always be DOWNLOAD
FIELD014 is client name and should always be LOGIS

Response (JSON): Array of upload table objects UPLOAD_XC_response.txt (see mapping section for fields values)

Get Sales Order Upload Confirmation

Description: Returns array of Sales Order numbers, which has been shipped based on XC records. Is not currently used.

Postman example	**Request type**	**URL**
Sales Order Upload Confirmation	GET	{{BaseUrl}}:12520/odata/UPLOADCollection

Query parameters: $filter=FIELD001 eq 'XC' and FIELD009 eq ' UPLOADED' and FIELD014 eq 'LOGIS' and UPLOADED eq '0'
If you are not familiar with this filtering syntax visit odata documentation section 5.1.1.1.11 Logical Operator Examples

Query example: {{BaseUrl}}:12520/odata/UPLOADCollection?$filter=FIELD001 eq 'XC' and FIELD009 eq 'UPLOADED' and FIELD014 eq 'LOGIS' and UPLOADED eq '0'

FIELD001 is type of record and it should always be XC
FIELD009 is type of status and should always be UPLOADED
FIELD014 is client name and should always be LOGIS

Response (JSON): same as in Sales Order Download Confirmation

Get Sales Order Pick Confirmation

Description: Returns array of all data for Sales Orders picking/shipping confirmations based on PC records

Postman example	**Request type**	**URL**
Sales Order Pick Confirmation	GET	{{BaseUrl}}:12520/odata/UPLOADCollection

Query parameters: $filter=FIELD001 eq 'PC' and FIELD056 eq ‘LOGIS’ and UPLOADED eq '0'

If you are not familiar with this filtering syntax visit odata documentation section 5.1.1.1.11 Logical Operator Examples

Query example: {{BaseUrl}}:12520/odata/UPLOADCollection?$filter=FIELD001 eq 'PC' and FIELD056 eq ‘LOGIS’ and UPLOADED eq '0'FIELD001 is type of record and it should always be PC

FIELD056 is client name and should always be ‘LOGIS’

UPLOADED is field with current status of that record

Response (JSON): Same structure as for XC records, but values in fields are different, (see mapping section for fields values)

Set Sales Order Pick Confirmation

Description: Updates PC records as read (processed by integration) for future deletion and to avoid duplicate reading of Picking/Shipping confirmation by Integration. Sets field uploaded to ‘1’.

Postman example	**Request type**	**URL**
Get Sales Order Pick Confirmation	PUT	{{BaseUrl}}:12520/odata/UPLOADCollection

Query: (guid'9E897FB6-0A2C-EC11-8D51-00155D016E35') RowId for row that should be updated

Query example: {{BaseUrl}}:12520/odata/UPLOADCollection(guid'9E897FB6-0A2C-EC11-8D51-00155D016E35')

Body (JSON): body should contain a single record that you get from (Sales Order Pick Confirmation) with modified field UPLOADED set to ‘1’.

Appropriate XC and PC records deletion

TBD